

Trace2Ground: Scenario, Disambiguation, and Relation Traces for Web GUI Grounding

Ruozhen He Zhongnan Qu Sheng Zhang Mesut Erhan Unal Zhuowen Tu
Srikar Appalaraju Jianbo Yuan Shabnam Ghadar Haofu Liao
AWS Agentic AI

Abstract

We present **Trace2Ground**, a large-scale web GUI grounding dataset and benchmark built by parsing HTML/DOM to obtain candidate interface elements and using large language models to simulate interaction traces and natural-language instructions. Existing datasets primarily target instructions solvable by matching the target element’s visible text, while real-world GUI tasks may require grounding under user scenarios, duplicate candidates, and relational descriptions. Trace2Ground targets these under-explored stress regimes through three curated splits: **PragmaMind**, where instructions describe user scenarios without naming UI elements; **ContextTwin**, where multiple elements share identical visible text and the correct target must be selected using interaction context; and **RelateLoc**, where the target is specified via spatial or structural relations to nearby anchors. We construct the dataset via an automated pipeline that extracts candidates from HTML/DOM structure, synthesizes traces and instructions with LLMs, and applies multi-view visual checks. Strong specialist GUI grounders underperform on these regimes despite reaching around 90% on ScreenSpot, exposing a gap not addressed by existing GUI training data. Standard supervised fine-tuning on Trace2Ground improves the base model’s performance on our curated regimes by 42.3% on average and yields zero-shot transfer to standard GUI benchmarks, including a 9.8% average improvement on the non-web ScreenSpot-Pro benchmark across six desktop domains.

1 Introduction

Graphical user interface (GUI) automation promises broad impact across the web, desktop software, and operating systems (Ye et al., 2025; Ma et al., 2024; Lai et al., 2024). Recent systems powered by large vision-and-language models (LVLMs) can follow natural-language

instructions to navigate pages, fill forms, and execute multi-step workflows (Lu et al., 2025a; Yan et al., 2023; Chen et al., 2025), enabling applications in assistance, testing, and accessibility. A core capability underlying these systems is *GUI grounding*: mapping an instruction and the current screen state to a concrete action on a specific interface element. Yet strong specialist GUI grounders that reach near-saturated accuracy on standard benchmarks degrade sharply on three common interaction patterns: instructions that describe scenarios without naming UI elements, screens with multiple elements sharing identical text, and targets specified through spatial relations to nearby anchors. These patterns are pervasive in real interfaces but rarely isolated by existing training data or evaluation.

Most existing GUI grounding datasets emphasize instructions that name the target element directly or match its visible text (Tab. 1). Large-scale corpora such as OS-Atlas (Wu et al., 2024) and ScreenSpot (Cheng et al., 2024) extract element references from HTML metadata or accessibility trees; LLM-augmented datasets like AutoGUI (Li et al., 2025a) and Aria-UI (Yang et al., 2025) add functional or history context but still produce instructions in which the target’s visible text appears verbatim in 90-99% of examples. Such data supports strong localization and text matching but does not stress three common interaction patterns: (i) users often express goals without naming UI elements (e.g., “turn on dark mode”), (ii) interfaces regularly contain multiple identical-text elements that require context to disambiguate (e.g., repeated “Edit” or “Log In” buttons), and (iii) users specify targets relationally (e.g., “the checkbox below Shipping Address”) rather than by exact identifiers. Under these conditions, models trained on visible text matching data may over-rely on surface cues and degrade sharply under ambiguity.

We introduce **Trace2Ground**, a large-scale

Table 1: **Comparison with related GUI grounding datasets.** *Avg Len*: average instruction length in words. *Vocab*: number of unique words. *Txt Ov.*: percentage of instructions containing the target element’s visible text verbatim. [†]UGround’s released parquet does not include target element text.

Work	Train	Test	Platf.	Instruction Style	Avg Len	Vocab	Txt Ov.
OS-Atlas (Wu et al., 2024)	2.3M	✗	W, M, D	Element references	1.9	12,381	94.0
ScreenSpot (Cheng et al., 2024)	1M	1.2k	W, M, D	Element references	3.4	1,129	30.3
ShowUI (Lin et al., 2025)	256k	✗	W, M, D	Element references + navigation history	6.2	5,535	99.6
AutoGUI (Li et al., 2025a)	702k	2k	W, M	Element references + functional state changes	23.5	4,003	26.5
Aria-UI (Yang et al., 2025)	4.2M	✗	W, M, D	Element references + action history	6.2	4,461	2.0
UGround (Gou et al., 2024)	10M	✗	W, M	Element references + spatial/positional cues	20.5	21,455	– [†]
Trace2Ground (Ours)	908k	41k	W	Scenarios, disambiguation, and relational	32.4	39,471	29.8

dataset and benchmark that explicitly curates grounding supervision for these three patterns through three splits: **PragmaMind** provides scenario-driven instructions that describe a user goal without naming any UI element; **ContextTwin** contains screens with multiple identical-text candidates, paired with context-conditioned descriptions that specify which instance is intended; and **RelateLoc** uses relational descriptions that identify targets via spatial or structural constraints to nearby anchors. Together, these splits provide training and evaluation data where success requires handling scenario-driven goals, duplicate-text disambiguation, and relational localization. These conditions occur frequently in practice but are rarely isolated by construction in existing data.

To scale beyond manual labeling, we build Trace2Ground from web pages (HTML/DOM) and LLM-synthesized instructions through an automated pipeline (Fig. 1). We parse the HTML/DOM structure to extract candidate interface elements and surface challenging configurations such as duplicate-text groups and relation-rich neighborhoods. For each split, we run a *small-scale refinement loop* in which LLMs synthesize initial instructions, human annotators audit the outputs, and feedback iteratively improves the generation prompts. Once prompts pass audit, we apply them at large scale, followed by an LLM-based filtering step and a final human audit that remove samples with verbatim target leakage, ambiguous instructions, or incorrect grounding labels.

Training on Trace2Ground substantially improves performance on all three regimes and transfers to established benchmarks, including out-of-distribution settings such as ScreenSpot-Pro across

six desktop domains. In summary, we make the following contributions:

- We introduce Trace2Ground, a large-scale web GUI grounding dataset and benchmark with three splits that isolate scenario-driven goals (PragmaMind), duplicate-text disambiguation (ContextTwin), and relational localization (RelateLoc).
- We propose an automated construction pipeline that parses HTML/DOM to extract candidate elements and challenging configurations, synthesizes instructions with LLMs, and applies multi-view visual checks together.
- We show that strong off-the-shelf grounders that exceed 90% on ScreenSpot drop to 23–61% on Trace2Ground regimes, and that standard SFT on Trace2Ground closes this gap by 42.3% on average while improving non-web ScreenSpot-Pro performance by 9.8%.

2 Related Work

Task-Completion Benchmarks. End-to-end benchmarks measure whether an agent reaches an intended goal state on realistic interfaces. Mind2Web (Deng et al., 2023) provides human-annotated trajectories on static snapshots, while WebLINX (Lù et al., 2024) pairs large-scale navigation trajectories with synthetic and human instructions. VisualWebArena (Zhou et al., 2023; Koh et al., 2024) and WebVoyager (He et al., 2024) evaluate agents on containerized and long-horizon live websites; Online-Mind2Web (Xue et al., 2025) additionally exposes brittleness under layout drift. AgentRek (Xu et al., 2024) synthesizes

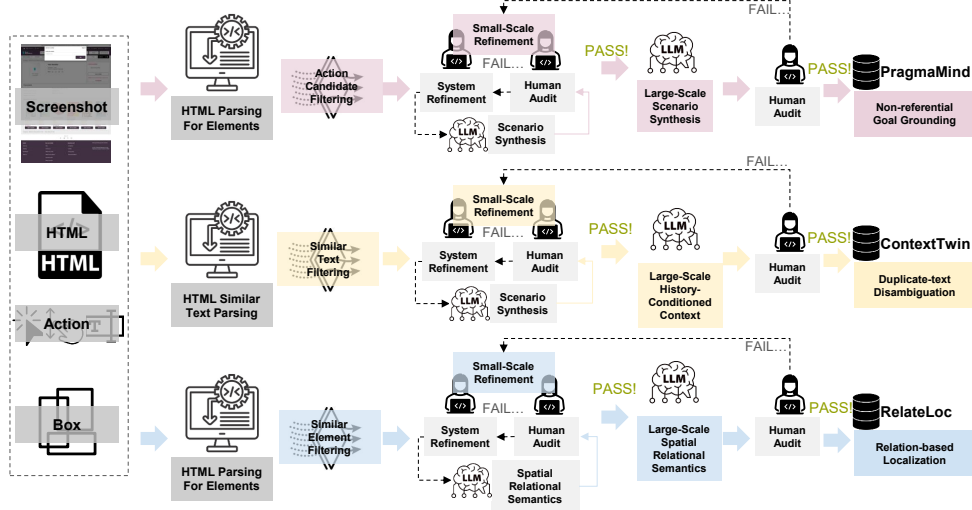


Figure 1: **Trace2Ground pipeline.** For each regime, we first parse web pages (HTML/DOM) to extract candidate elements and surface challenging configurations (e.g., duplicate-text groups or relation-rich neighborhoods). We then run a *small-scale refinement loop*: LLMs synthesize initial instructions, which are reviewed by human annotators and fed back into system refinement to iteratively improve prompt quality. Once prompts pass audit, we apply them at *large scale* to synthesize the full dataset. A final human audit stage ensures the large-scale quality, yielding three common interactive regimes: PragmaMind, ContextTwin, and RelateLoc.

multistep trajectories from online tutorials, and OSWorld (Xie et al., 2024) extends end-to-end execution to real desktop applications. These benchmarks primarily report aggregate goal success and provide limited visibility into where grounding breaks down.

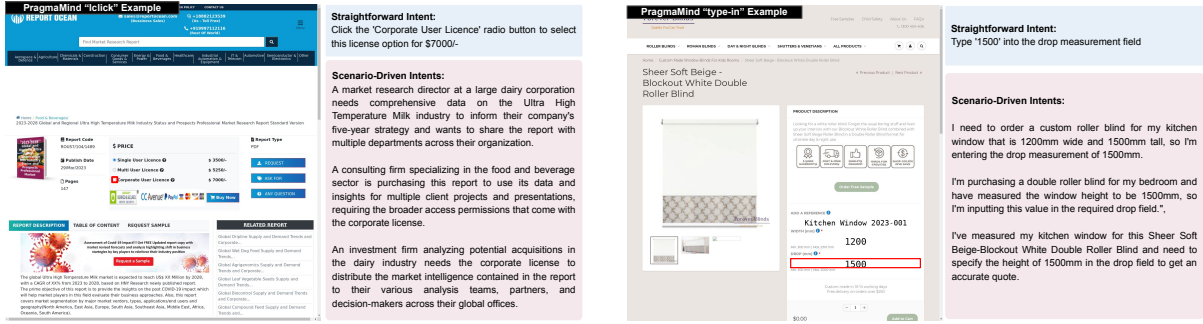
GUI Grounding Benchmarks. A separate line of work targets screen understanding and element localization. GUI-360 (Mu et al., 2025) decomposes ability into grounding, UI parsing, and action prediction at the scale of millions of steps; ScreenSuite (Roucher and Mahla, 2025) unifies thirteen datasets under a shared protocol. ScreenSpot (Cheng et al., 2024) and ScreenSpot-Pro (Li et al., 2025b) measure fine-grained element localization, and OS-Atlas (Wu et al., 2024) maps large-scale screen data into structured representations. UI-E2I-Synth (Liu et al., 2025) studies implicit instructions and distributional factors such as element-to-screen ratio. VenusBench-GD (Zhou et al., 2025) broadens grounding tasks across platforms with Reasoning and Refusal subsets, and WinClick (Hui et al., 2025) targets Windows screenshots. Among these, only UI-E2I-Synth partially overlaps with our PragmaMind regime, and none separately measures identical-text disambiguation or relational localization.

Browsing and Information-Seeking Benchmarks. A third class of benchmarks evaluates long-horizon web navigation for information retrieval. BrowseComp (Wei et al., 2025) measures persistent browsing for difficult factual queries, and

MM-BrowseComp (Li et al., 2025c) extends this to multimodal content. These benchmarks emphasize external information acquisition rather than interface-level grounding and are largely orthogonal to our setting.

3 Trace2Ground Dataset

Trace2Ground is a large-scale web GUI grounding dataset and benchmark that curates supervision for three common interaction patterns: scenario-driven goals, identical-text disambiguation, and relational localization. It is built from web pages by parsing HTML/DOM to obtain candidate interface elements and using LLMs to synthesize natural-language instructions and interaction context (Fig. 1). Each sample consists of a screenshot x_{img} , an HTML/DOM tree x_{html} , a natural-language instruction q , and a target grounding label $y = (a, b)$, where a is the intended action (e.g., lclick, type-in) and b is either a DOM element id or a normalized bounding box $\in [0, 1]^4$. Trace2Ground organizes the data into three splits, each described in detail below: **PragmaMind** (§3.1) provides scenario-driven instructions that describe a user goal without naming any UI element; **ContextTwin** (§3.2) contains screens with multiple elements sharing identical visible text, paired with context-conditioned instructions that specify which instance to act on; and **RelateLoc** (§3.3) uses relational descriptions tied to nearby anchors.



(a) lclick: the instruction describes a user goal, and the model must identify which clickable element to act on.

(b) type-in: the instruction embeds the text to type within a scenario, and the model must identify the correct input field.

Figure 2: **Examples from PragmaMind.** Scenario-driven instructions describe a user goal without naming the target UI element, requiring the model to ground the goal to an action-target pair from page context.

3.1 PragmaMind: Scenario Grounding

PragmaMind targets a common practical setting: users often describe what they want to accomplish without naming a specific UI element. Given a GUI state (x_{img}, x_{html}) , we extract interactive candidates from the DOM and use an LLM to synthesize scenario-driven instructions q that describe the user’s goal, requiring the model to map q to an action-target pair (a, y) using page context.

Click actions. For lclick actions, we identify webpages with multiple clickable elements (buttons, checkboxes, etc.) through DOM attributes and visibility checks. Each target y is paired with two visual inputs: a full-page screenshot capturing the overall layout, and a cropped region highlighting the interaction area. An LLM rewrites the action into a scenario-driven instruction q that would naturally lead a user to click y without mentioning the element’s name or visible text. A built-in LLM judge then filters samples that (i) leak the target’s visible text, (ii) rely on style-specific icon descriptions unlikely to generalize, or (iii) lack a clear actionable goal.

Type-in actions. For type-in actions, we locate input fields from the DOM and obtain the intended text from the synthesized specification. To enhance visual realism, we optionally render partial text into neighboring fields on x_{img} , simulating a user partway through a form. The LLM then generates an instruction q that embeds the desired text entry within a plausible scenario implicitly. We apply the same judge step to remove outputs that leak field names, contain mismatched content, or are solvable via trivial keyword matching.

3.2 ContextTwin: Contextual Disambiguation

Many interfaces contain repeated labels, such as multiple “Edit” buttons in a settings panel, repeated

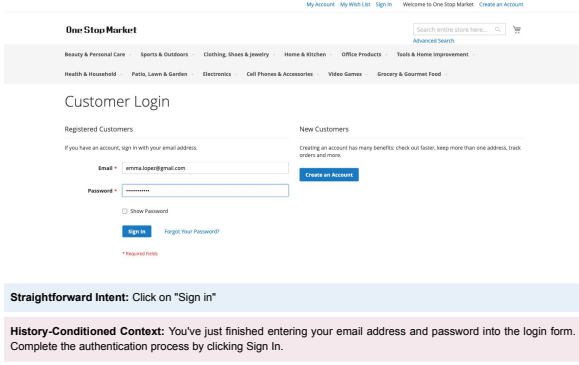
product names on a shopping page, or several “Reply” links in an email thread. In these cases, the target cannot be identified from visible text alone; users rely on interaction context (what step they are on, what just changed) and local layout cues to select the correct instance. ContextTwin isolates this setting by constructing examples where multiple candidates share identical visible text and the correct target must be selected through context rather than text.

Given a GUI state (x_{img}, x_{html}) , we parse the DOM to extract textual attributes and group elements that share the same normalized string, retaining only groups $C = \{e_1, \dots, e_K\}$ with $K \geq 2$ that satisfy visibility and interactability constraints. For each group, we designate one element $y \in C$ as the target and sample at least one additional element $d \in C$ as a distractor, recording normalized bounding boxes for validation and to produce a zoomed crop enclosing the competing instances.

An LLM is then provided with the full-page screenshot with target/distractor boxes, a zoomed crop containing the competing instances, and the original action description. The model first produces brief region summaries and high-level layout cues (e.g., section of the page, surrounding content), then rewrites the action into a natural instruction that would lead a user to act on y without referencing the shared visible text. When applicable, the instruction incorporates a short history summary h describing the immediately preceding step or page state change.

3.3 RelateLoc: Relation-Based Localization

Interfaces frequently contain multiple plausible targets with overlapping or repeated labels, such as, a few comment boxes on the same page, “Add to Cart” buttons in a product grid, or repeated “Download”



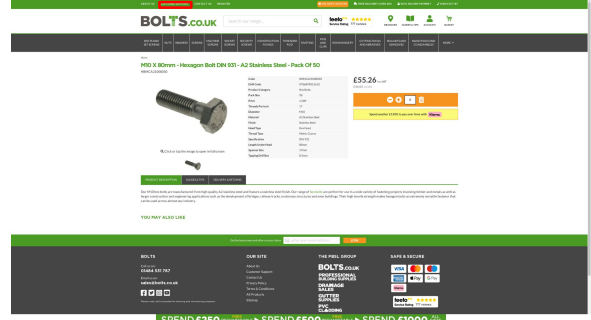
(a) ContextTwin: multiple elements share identical visible text, and the instruction uses interaction context (e.g., a history summary) to specify the intended instance.

Figure 3: **Examples from ContextTwin and RelateLoc.** Both regimes require grounding beyond visible-text matching: ContextTwin through context-conditioned disambiguation among identical-text candidates, and RelateLoc through relational reasoning over nearby anchors.

links for different formats. In these cases, users naturally specify a target using spatial or structural references (e.g., the download link next to the PDF option) rather than the label alone. RelateLoc isolates this setting by constructing examples where the correct element is identified through relations to nearby elements in the layout.

Given a GUI state $(x_{\text{img}}, x_{\text{html}})$, we parse the DOM to extract visible, interactable candidates and identify contexts where similar or identical strings appear multiple times. For each target y , we analyze nearby plausible distractors from the same structural area (e.g., other buttons in the same toolbar, links in the same sidebar, or items within the same list), and retain their bounding boxes to increase disambiguation difficulty. RelateLoc allows the target’s visible text to appear in the instruction (e.g., “Edit” or “Save”), but requires additional relational cues tied to surrounding anchors, so that successful localization cannot be achieved by text matching alone.

Relational intent generation. We provide an LLM with three inputs: a full-page screenshot with the target and distractors overlaid in contrasting colors, a zoomed crop enclosing the competing candidates for precise reading, and the original action description. The model first produces concise summaries of the candidate regions and their approximate positions (e.g., within the header, sidebar, or a specific section), then rewrites the action into a relation-based instruction q that a user could follow to act on y . The instruction may include the target’s text but must also contain informative spatial or structural relations to nearby anchors. Examples



(b) RelateLoc: the instruction specifies the target through spatial or structural relations to nearby anchors (e.g., position within the navigation bar).

include “the first link under the ‘Pages’ heading in the right sidebar”, “the Save button directly below the comment box”, and “the Edit icon in the toolbar next to the Share button”. We filter generations that reduce to bare name matches without meaningful relational information, and verify that the final relational description uniquely identifies the target among the retained distractors.

4 Experimental Setup

Quality Assurance. Inspired by prior model-in-the-loop pipelines (Kirillov et al., 2023; Wang et al., 2023; Xu et al., 2025; Ravi et al., 2024), our pipeline (Fig. 1) balances large-scale autonomous annotation with manual quality control. We first run a *small-scale refinement loop* on $\sim 1\%$ of the final data scale: LLMs synthesize initial instructions, human annotators audit the outputs, and feedback iteratively improves the generation prompts. This refinement is critical: initial prompts produce only $\sim 60\%$ acceptable outputs, and successive revisions raise the pass rate to $\geq 95\%$. Once prompts pass audit, we apply them at large scale, then filter through DOM validity checks, multi-view visual verification, and bounding-box consistency, reducing approximately 3M raw web-page candidates to 1.2M. A built-in LLM judge step further rejects samples that (i) leak the target’s visible text, (ii) rely on style-specific icon descriptions unlikely to generalize, or (iii) lack a clear actionable goal, removing 20–25% of candidates per regime and yielding the final 907,684 training examples.

We quantify the judge’s contribution by measuring annotation accuracy on stratified samples (100

Table 2: Success rate (%) on the Screenshot-pro benchmark across six application domains. We report separate performance for text- and icon-based referring expressions and aggregate averages (Avg./Total). GPT-4o and Claude results are from (Luo et al., 2025).

	CAD		Creative		Dev		Office		OS		Scientific		Avg.		
	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Total
GPT-4o (Hurst et al., 2024)	2.0	0.0	1.0	0.0	1.3	0.0	1.1	0.0	0.0	0.0	2.1	0.0	1.3	0.0	0.8
Claude Computer Use (Anthropic, 2024)	14.5	3.7	25.9	3.4	22.0	3.9	30.1	16.3	11.0	4.5	33.9	15.8	23.4	4.0	18.9
Qwen2.5-VL 3B (Bai et al., 2025)	9.1	7.3	26.8	2.1	22.1	1.4	33.9	15.1	10.3	1.1	38.2	7.3	23.6	4.4	16.3
Qwen2.5-VL 7B (Bai et al., 2025)	16.8	1.6	35.9	7.7	46.8	4.1	52.5	20.8	37.4	6.7	49.3	7.3	38.9	7.1	26.8
CogAgent 18B (Hong et al., 2024)	7.1	3.1	9.6	0.0	14.9	0.7	13.0	0.0	5.6	0.0	22.2	1.8	12.0	0.8	7.7
Aria-UI (Yang et al., 2025)	7.6	1.6	23.7	2.1	16.2	0.0	20.3	1.9	4.7	0.0	27.1	6.4	17.1	2.0	11.3
OS-Atlas 7B (Wu et al., 2024)	12.2	4.7	28.8	2.8	33.1	1.4	33.9	5.7	27.1	4.5	37.5	7.3	28.1	4.0	18.9
ShowUI 2B (Lin et al., 2025)	2.5	0.0	9.1	0.0	16.9	1.4	15.3	7.5	10.3	2.2	13.2	7.3	10.8	2.6	7.7
UGround 7B (Gou et al., 2024)	14.2	1.6	27.3	2.8	26.6	2.1	31.6	11.3	17.8	0.0	31.9	2.7	25.0	2.8	16.5
UGround-v1 7B (Gou et al., 2024)	15.8	1.2	47.5	9.7	51.9	2.8	60.5	13.2	38.3	7.9	57.6	14.5	44.6	8.1	30.7
UI-R1 3B (Lu et al., 2025b)	11.2	6.3	27.3	3.5	22.7	4.1	32.2	11.3	13.1	4.5	42.4	11.8	24.9	6.3	17.8
GUI-R1 3B (Luo et al., 2025)	26.4	7.8	40.9	5.6	33.8	4.8	53.6	17.0	28.1	5.6	61.8	17.3	40.8	8.8	28.6
GUI-R1 7B (Luo et al., 2025)	23.9	6.3	38.9	8.4	49.4	4.8	58.7	26.4	42.1	16.9	55.6	11.8	43.9	10.8	31.3
UI-TARS 2B (Qin et al., 2025)	17.8	4.7	42.9	6.3	47.4	4.1	50.3	17.0	21.5	5.6	56.9	17.3	39.6	8.4	27.7
UI-TARS 7B (Qin et al., 2025)	20.8	9.4	50.0	9.1	58.4	12.4	63.3	20.8	30.8	16.9	63.9	31.8	47.8	16.2	35.7
UI-TARS 72B (Qin et al., 2025)	18.8	12.5	57.1	15.4	62.9	17.2	63.3	26.4	42.1	15.7	64.6	20.9	50.9	17.5	38.1
Trace2Ground 3B	27.4	12.5	46.5	14.7	50.7	15.9	57.6	37.7	30.8	15.7	58.3	25.5	45.3	18.9	35.2
Trace2Ground 7B	33.0	12.5	43.9	16.1	59.7	15.9	57.1	39.6	30.8	18.0	58.3	19.1	47.3	18.6	36.3

Table 3: Success rate (%) on the ScreenSpot benchmark. We report performance on web-based text and icon referring expressions, plus the overall average.

Model	Web-Text	Web-Icon	Avg.
GPT-4 (Achiam et al., 2023)	9.2	8.8	9.0
GPT-4o (Hurst et al., 2024)	12.2	7.8	10.0
Qwen2.5-VL 3B (Bai et al., 2025)	60.8	43.5	52.2
Qwen2.5-VL 7B (Bai et al., 2025)	86.9	65.1	76.0
SeeClick (Cheng et al., 2024)	55.6	32.5	44.1
OS-Atlas 4B (Wu et al., 2024)	82.6	63.1	72.9
OS-Atlas 7B (Wu et al., 2024)	90.8	74.2	82.5
CogAgent 18B (Hong et al., 2024)	70.4	28.6	49.5
UGround 7B (Gou et al., 2024)	80.4	70.4	75.4
Trace2Ground 3B	81.3	76.2	78.8
Trace2Ground 7B	83.0	85.4	84.2

instances per regime, three expert annotators, majority vote) before and after filtering (Tab. 6). The judge improves accuracy by 18–20 points across regimes, with the larger gains on ContextTwin and RelateLoc reflecting the higher difficulty of generating unambiguous instructions when targets share text or are specified relationally.

Training Data. We fine-tune on four public GUI datasets as baselines and then combined with Trace2Ground. Trace2Ground training splits contain 277,948 examples from PragmaMind-click, 310,726 from PragmaMind-type-in, 160,010 from ContextTwin, and 159,000 from RelateLoc. Mind2Web (Deng et al., 2023) is a web automation dataset with 2,350 language-based tasks across 137 websites, where each example contains a high-level goal and crowdsourced action trajectories. WebLINX (Lù et al., 2024) provides real-world website navigation data with multi-turn dialogue-based instruction following. OS-

Table 4: Performance on Trace2Ground diagnostic regimes (Step SR, %). Qwen2.5-VL* denotes the public-data SFT baseline.

Model	PM-Click	PM-Type	CxtTwin	ReLoc
ShowUI-2B	3.9	12.2	4.5	4.4
Aria-UI	8.2	2.2	7.9	1.5
Qwen2.5-VL-7B	36.9	25.1	22.2	23.0
OS-Atlas-7B	14.7	52.9	26.1	7.6
UGround-V1-7B	25.2	78.4	14.9	55.3
UI-TARS-7B	23.0	76.0	56.6	61.0
Qwen2.5-VL* 3B	46.4	14.2	36.0	39.9
Trace2Ground 3B	94.3	90.5	76.6	72.8
Qwen2.5-VL* 7B	48.4	31.0	45.3	45.4
Trace2Ground 7B	94.2	93.9	77.2	74.4

Atlas (Wu et al., 2024) is a single-step, cross-platform GUI grounding corpus with over 13M annotated elements across mobile, desktop, and web. RICO-WidgetCaptioning (Deka et al., 2017) extends the RICO mobile UI dataset (~66k screens, 3M elements) with widget captions and bounding boxes. We regenerate instructions with Claude 3.7 (Anthropic, 2024) for Mind2Web and WebLINX to align instruction style across sources.

Evaluation Data. We evaluate on the held-out Trace2Ground test splits along with three external benchmarks. ScreenSpot and ScreenSpot-Pro (Cheng et al., 2024; Li et al., 2025b) are single-step grounding benchmarks with text- and icon-based referring expressions, covering web screenshots and six non-web desktop domains respectively. Visual Web Arena (Koh et al., 2024) is a multi-step web interaction benchmark requiring action grounding across diverse layouts. We also use the held-out test split of Mind2Web for both

Table 5: **Ablation on data mixtures for a 7B backbone on ScreenSpot-Pro.** Each configuration uses the same total number of training samples; when a regime is excluded, it is replaced with an equal number of samples from the public UI grounding data (Mind2Web, WebLINX, OS-Atlas, RICO). We report success rate (%) across six desktop domains, split by text- vs. icon-based grounding, along with averaged performance (Avg.) and overall Total.

	CAD		Creative		Dev		Office		OS		Scientific		Avg.		Total
	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	
Qwen2.5-VL 7B (Bai et al., 2025)	16.8	1.6	35.9	7.7	46.8	4.1	52.5	20.8	37.4	6.7	49.3	7.3	38.9	7.1	26.8
Base (Qwen2.5-VL* 7B)	26.4	7.8	39.9	9.8	54.5	17.9	55.4	26.4	27.1	13.5	56.3	18.2	43.3	15.1	32.5
Base+PragmaMind	28.9	12.5	40.9	13.9	53.8	18.6	58.1	28.3	27.1	16.8	53.4	20.0	44.0	17.7	33.9
Base+ContextTwin	26.9	14.0	34.3	13.1	51.2	19.3	54.2	32.1	30.8	17.9	54.8	20.0	41.7	18.3	32.8
Base+RelateLoc	29.4	12.5	42.4	13.3	57.1	14.5	59.9	34.0	29.9	11.2	61.1	15.5	46.7	15.4	34.7
Trace2Ground 7B	33.0	12.5	43.9	16.1	59.7	15.9	57.1	39.6	30.8	18.0	58.3	19.1	47.3	18.6	36.3

Table 6: Effect of built-in LLM filtering on annotation accuracy. We report the average human-verified accuracy (%) on 100 stratified samples (3 times) per regime, with and without the built-in LLM judge, along with the rejection rate.

Regime	Before	Rejection	After	Δ
PragmaMind	78	20%	96	+18
RelateLoc	70	22%	90	+20
ContextTwin	66	25%	86	+20

single-step and multi-step evaluations.

Metrics. We report Step Success Rate (Step SR), the fraction of steps where the center of the predicted bounding box falls within the ground-truth element: $\text{Step_SR} = \frac{1}{N} \sum_{i=1}^N \mathbb{1}\{c_i^{\text{pred}} \in b_i^{\text{gt}}\}$. Operation F1, which measures action-type prediction accuracy, is near-saturated for all models we evaluate; we report it in App. E.

Setup. We use Claude-3.7 (Anthropic, 2024) as the instruction generation LLM and Qwen2.5-VL (Bai et al., 2025) as the VLM grounder backbone. Training uses standard autoregressive next-token prediction on a serialized (a, b^*) target; full hyperparameters and training details are in App. D.

5 Experimental Results

Zero-shot performance on out-of-distribution standard benchmarks. Tab. 2 and Tab. 3 report zero-shot performance on ScreenSpot and ScreenSpot-Pro. Although Trace2Ground is built exclusively from web pages, fine-tuning on it transfers to all six non-web domains in ScreenSpot-Pro, indicating robustness under layout and visual-style shift. Trace2Ground 7B achieves the highest icon-based accuracy on ScreenSpot-Pro across all reported models (18.9, vs. 17.5 for UI-TARS 72B) and the highest icon-based accuracy on ScreenSpot (85.4) among open 7B-scale models, suggesting that the supervision particularly improves grounding under visually compact targets where text matching is least available. On overall ScreenSpot-

Pro accuracy, Trace2Ground 7B (36.3) is competitive with UI-TARS 72B (38.1) despite using over an order of magnitude less training data.

Performance on Trace2Ground. Tab. 4 evaluates six off-the-shelf UI grounders, a general-purpose VLM, the public-data SFT baseline (Qwen2.5-VL*), and our Trace2Ground-trained models on the three regimes. Models that perform strongly on ScreenSpot fall short when goals are indirect, candidates share identical text, or targets are specified relationally. The within-PragmaMind asymmetry is especially telling. UI-TARS-7B reaches 76.0% on PM-Type but 23.0% on PM-Click, mirroring the text-overlap difference between the two splits (4.1% vs. 58.5% of instructions contain the target text verbatim). While the public-data SFT baseline (Qwen2.5-VL*) improves over the base VLM, it still underperforms substantially across all four splits. Fine-tuning with Trace2Ground closes the gap uniformly, with consistent gains at both 3B and 7B scale and the largest improvements on the regimes where baselines were weakest.

Per-regime contributions under matched data budget. Tab. 5 analyzes the contribution of each regime under a controlled-scale setting: all configurations use the same total number of training samples, and when a Trace2Ground regime is excluded it is replaced with an equal number of examples from the public UI grounding mixture (Mind2Web, WebLINX, OS-Atlas, RICO). This isolates the nature of the supervision from data scale. Each individual regime improves over the public-data baseline (Qwen2.5-VL* 7B, 26.8% Total), with RelateLoc the strongest single regime on text-based grounding (47.3% Text avg) and ContextTwin the strongest on icon-based grounding (18.6% Icon avg). Combining all three yields the best overall performance (36.3% Total, +9.5 over Base), with

Table 7: Generalization of Trace2Ground to a Phi-3 (Abdin et al., 2024) backbone on PragmaMind, ContextTwin, and RelateLoc. Step SR and operation-level F1 are reported in %. Phi-3* denotes the base model before training on Trace2Ground. Bottom: Phi-3 backbone performance across six desktop UI domains on Trace2Ground. We report success rate (%) for text and icon referring expressions in each domain.

Model	PragmaMind (lclick)		PragmaMind (type-in)		ContextTwin		RelateLoc	
	Step SR	Op. F1	Step SR	Op. F1	Step SR	Op. F1	Step SR	Op. F1
Phi-3*	11.4	94.5	23.6	62.8	27.1	97.9	29.7	99.6
Trace2Ground Phi-3	77.7	100.0	73.2	93.5	70.5	99.9	66.7	99.9

Model	CAD		Creative		Dev		Office		OS		Scientific	
	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon
Phi-3*	4.7	11.3	34.8	2.2	38.3	2.8	45.8	20.8	27.1	5.6	41.67	8.2
Trace2Ground Phi-3	14.7	6.3	40.4	3.5	49.4	4.8	48.0	22.6	24.3	6.7	45.8	9.1

Table 8: Mixed-regime evaluation. 5,000 samples drawn equally from PragmaMind-Click, ContextTwin, and RelateLoc and randomly interleaved at test time.

Model	Overall	PM-Click	CxtTwin	ReLoc
OS-Atlas-7B	19.3	12.0	22.6	23.2
UGround-V1-7B	45.7	23.2	57.4	56.4
UI-TARS-7B	46.8	21.7	56.2	62.5
Trace2Ground 7B	81.1	93.5	75.6	74.1

the largest gains on icon-based grounding where surface text matching is least available. These results suggest that the three regimes provide complementary supervision: each specializes in a different aspect of robust grounding, and their combination transfers across diverse UI domains.

Robustness under mixed-regime evaluation. Real-world GUI interaction does not announce which regime an instruction belongs to. We construct a mixed-regime test split of 5,000 examples drawn equally from PragmaMind-Click, ContextTwin, and RelateLoc and randomly interleaved (Tab. 8). Trace2Ground-7B achieves 81.1% overall, closely matching its single-regime performance and showing that the model handles each type of instruction without an explicit regime label. Strong baselines remain bounded by their weakest regime. UI-TARS-7B reaches 46.8% and UGround-V1-7B 45.7%, both bottlenecked on PragmaMind-Click, confirming that the three regimes test complementary capabilities that Trace2Ground supervision teaches simultaneously.

Generalization to other backbones. To test whether Trace2Ground supervision is tied to a specific LVLM family, we instantiate it on Phi-3 (Abdin et al., 2024). Training with Trace2Ground

Table 9: Phi-3 (Abdin et al., 2024) backbone results on ScreenSpot, Visual Web Arena, and Mind2Web. We report success rate (SR, %) for text- and icon-based grounding on ScreenSpot and grounding accuracy on Visual Web Arena and Mind2Web evaluation subsets.

Model	ScreenSpot		VWA		Mind2Web
	Web-Text	Web-Icon			
Phi-3*	79.6	64.1	73.4		59.4
Trace2Ground Phi-3	80.0	67.0	99.9		66.4

transforms Phi-3* into a strong grounder: Step SR jumps from 11.4–29.7% to 66.7–77.7% across the four regimes (Tab. 7), mirroring the Qwen2.5-VL pattern. Trace2Ground-Phi-3 also transfers beyond the regimes, with 5–10 point gains on ScreenSpot-Pro text-based grounding and consistent boosts on icon-based grounding, plus substantial improvements on Visual Web Arena and Mind2Web (Tab. 9). The two-backbone result shows Trace2Ground supervision transfers across LVLM families.

Per-Category Error Analysis. To understand where Trace2Ground supervision helps most, we compare a public-data SFT baseline (Qwen2.5-VL*) against the same backbone fine-tuned with Trace2Ground on top of the public mixture. On ScreenSpot web-text, Trace2Ground improves overall accuracy from 83.0 to 87.8 (by +4.8 points). We further partition the test set by query type to identify which capabilities drive this gain.

The per-category breakdown reveals three patterns. First, queries where the target’s visible text is absent from the instruction improve by +5.1 points, consistent with Trace2Ground’s PragmaMind training. Second, small-element grounding (targets occupying less than 1% of screen area) improves by +9.1 points, the largest single category gain, reflect-

Table 10: **Failure analysis of Trace2Ground-7B on the Trace2Ground test set.** Categories are mutually exclusive, assigned in order of priority (Near miss $\rightarrow$ Tiny element $\rightarrow$ Type-in localization $\rightarrow$ Text matching $\rightarrow$ Intent understanding).

Failure category	% of failures
Near miss	41.5
Tiny element	39.3
Type-in localization	10.0
Text matching	1.9
Intent understanding	1.8

ing that Trace2Ground targets are predominantly small (App. C). Third, queries containing spatial or positional cues (“below”, “left of”, “next to”) improve by +4.0 points, consistent with RelateLoc supervision. These category-level gains indicate that improvements stem from the specific capabilities Trace2Ground targets, not from generic accuracy improvements.

6 Failure Analysis

We analyze where Trace2Ground-7B still makes errors on the Trace2Ground test set. The model achieves 91.7% overall accuracy. We manually categorize the remaining failures into five categories based on the predicted bounding box, the target bounding box, and the instruction (Tab. 10).

Categories. We define five failure categories:

- **Near miss.** Predicted center falls outside the target box but within $2\times$ the target’s bounding box. The model has localized to the correct region but missed the exact element.
- **Tiny element.** Target occupies less than 0.5% of the screen area. Many of these targets are also near-misses, but we assign them to this category first to surface the size dependence.
- **Type-in localization.** Misidentified input field on PragmaMind-Type, typically selecting a neighboring field within the same form.
- **Text matching.** Predicted element shares no textual relation to the target despite the instruction containing the target’s visible text (relevant primarily for ContextTwin and RelateLoc).
- **Intent understanding.** Predicted element is unrelated to the user’s goal, indicating the model did not correctly interpret the scenario or relational description.

Interpretation. Two observations stand out. First, roughly 81% of failures (near miss + tiny element) are *perceptual* rather than semantic: the model identifies the right region of the screen but misses the exact element, typically on small targets. This aligns with the target size distribution in Fig. 10, where Trace2Ground targets cluster well below 0.5% of screen area. Second, only 3.7% of failures (text matching + intent understanding) are semantic, suggesting that scenario, disambiguation, and relational understanding are largely solved at this scale. The remaining headroom lies in higher-resolution visual perception rather than additional instruction-level supervision.

7 Conclusion

We introduced Trace2Ground, a large-scale web GUI grounding dataset and benchmark targeting three common interaction patterns: scenario-driven goals (PragmaMind), identical-text disambiguation (ContextTwin), and relational localization (RelateLoc). Trace2Ground is constructed automatically from web pages by parsing HTML/DOM for candidate elements and challenging configurations, and using LLMs to synthesize instructions with multi-view visual checks and leakage filtering. Strong specialist GUI grounders that exceed 90% on ScreenSpot drop to 23–61% on the three regimes. Standard supervised fine-tuning on Trace2Ground closes this gap and transfers to out-of-distribution benchmarks, including a 9.8% average improvement on the non-web ScreenSpot-Pro. The same supervision generalizes across Qwen2.5-VL and Phi-3 backbones, indicating that targeted, web-only data can teach grounding capabilities that transfer broadly.

Limitations

Trace2Ground is built exclusively from web pages, which introduces a domain gap relative to native mobile and desktop interfaces. Mobile UIs in particular present distinct interaction patterns (swipe targets, bottom navigation bars) and sparser DOM structures that web-derived supervision does not fully cover. Extending Trace2Ground to mobile and desktop domains, leveraging accessibility trees from Android or Windows alongside our HTML/DOM pipeline, is a natural direction for future work. A second direction is moving beyond supervised fine-tuning: reinforcement learning on Trace2Ground could provide an alternative training signal for the same three regimes.

References

- Marah Abidin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, and 1 others. 2024. [Phi-3 technical report: A highly capable language model locally on your phone](#). *Preprint*, arXiv:2404.14219.
- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Anthropic. 2024. Developing a computer use model, October 22. <https://www.anthropic.com/news/developing-computer-use> [Accessed: 2025-11-14].
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.
- Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, and 1 others. 2025. Guicourse: From general vision language model to versatile gui agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21936–21959.
- Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. SeeClick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332.
- Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hirschman, Daniel Afegan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 845–854.
- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.
- Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*.
- 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, Weihang 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.
- Zheng Hui, Yinheng Li, Tianyi Chen, Colby Banbury, Kazuhito Koishida, and 1 others. 2025. Winclick: Gui grounding with multimodal large language models. *arXiv preprint arXiv:2503.04730*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, and 1 others. 2023. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026.
- Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*.
- Hanyu Lai, Xiao Liu, Iat Long Long, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and 1 others. 2024. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5295–5306.
- Hongxin Li, Jingfan Chen, Jingran Su, Yuntao Chen, Li Qing, and Zhaoxiang Zhang. 2025a. Autogui: Scaling gui grounding with automatic functionality annotations from llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10323–10358.
- Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. 2025b. Screenspot-pro: Gui grounding for professional high-resolution computer use. *arXiv preprint arXiv:2504.07981*.
- Shilong Li, Xingyuan Bu, Wenjie Wang, Jiaheng Liu, Jun Dong, Haoyang He, Hao Lu, Haozhe Zhang, Chenchen Jing, Zhen Li, Chuanhao Li, Jiayi Tian, Chenchen Zhang, Tianhao Peng, Yancheng He, Jiahao Gu, Yuanxing Zhang, Jian Yang, Ge Zhang, and 5 others. 2025c. [Mm-browsecomp: A comprehensive benchmark for multimodal browsing agents](#). *Preprint*, arXiv:2508.13186.
- Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2025. Showui:

- One vision-language-action model for gui visual agent. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19498–19508.
- Xinyi Liu, Xiaoyi Zhang, Ziyun Zhang, and Yan Lu. 2025. Ui-e2i-synth: Advancing gui grounding with large-scale instruction synthesis. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15668–15684.
- Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. 2025a. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22404–22414.
- Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*.
- Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanqing Xiong, and Hongsheng Li. 2025b. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*.
- Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. 2025. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*.
- Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9097–9110.
- Jian Mu, Chaoyun Zhang, Chiming Ni, Lu Wang, Bo Qiao, Kartik Mathur, Qianhui Wu, Yuhang Xie, Xiaojun Ma, Mengyu Zhou, Si Qin, Liqun Li, Yu Kang, Minghua Ma, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2025. *Gui-360°: A comprehensive dataset and benchmark for computer-using agents*. Preprint, arXiv:2511.04307.
- Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Uitars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.
- Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, and 1 others. 2024. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*.
- Aymeric Roucher and Amir Mahla. 2025. Screensuite: the most comprehensive benchmarking suite for gui agents. <https://github.com/huggingface/screensuite>.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khoshnab, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.
- Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. *Browsecomp: A simple yet challenging benchmark for browsing agents*. Preprint, arXiv:2504.12516.
- Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and 1 others. 2024. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*.
- Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. *Os-world: Benchmarking multimodal agents for open-ended tasks in real computer environments*. Preprint, arXiv:2404.07972.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2025. Wizardlm: Empowering large pre-trained language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.
- Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. 2024. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. *arXiv preprint arXiv:2412.09605*.
- Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. 2025. *An illusion of progress? assessing the current state of web agents*. In *Second Conference on Language Modeling*.
- An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, and 1 others. 2023. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*.
- Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2025. Aria-ui: Visual grounding for gui instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22418–22433.
- Yiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, and 1 others. 2025. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*.

Beitong Zhou, Zhexiong Huang, Yuan Guo, Zhangxuan Gu, Tianyu Xia, Zichen Luo, Fei Tang, Dehan Kong, Yanyi Shang, Suling Ou, and 1 others. 2025. Venusbench-gd: A comprehensive multi-platform gui benchmark for diverse grounding tasks. *arXiv preprint arXiv:2512.16501*.

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. *arXiv preprint arXiv:2307.13854*.

In this appendix, we provide reasons for choosing the web domain in Sec. A, additional data curation details in Sec. B, dataset statistical analysis in Sec. C, SFT details in Sec. D, operation F1 results in Sec. E, data examples in Sec. F, qualitative examples in Sec. G, and broader impact in Sec. H.

A On the Choice of Web Domain

Trace2Ground is built exclusively from web pages. This section explains why web pages are particularly well-suited to constructing the three regimes at scale, and why the choice does not limit the generality of the learned grounding skills.

A.1 Web Pages Enable Easy Curation

Each Trace2Ground regime depends on structural properties that are abundant and well-defined in web pages.

PragmaMind. Scenario-driven grounding requires pages with rich semantic context—enough situational information for a human to infer the correct action without seeing any UI label. Web pages provide this naturally: product descriptions, form workflows, and editorial content supply the surrounding context that makes scenario-driven instructions interpretable. The HTML/DOM tree further exposes element roles, enabling reliable candidate extraction and action-type supervision without manual annotation.

ContextTwin. Identical-text disambiguation requires interfaces with *naturally occurring* repeated labels. Web pages exhibit this pattern at high frequency: product grids repeat “Add to Cart”, navigation headers repeat “Edit” or “View”, and comment threads repeat “Reply” or “Like” across dozens of instances per page. Desktop applications and mobile apps tend to avoid label repetition within a single screen by design, making large-scale ContextTwin construction from those domains substantially harder. Our DOM-based grouping is directly

enabled by the structured, machine-readable nature of HTML.

RelateLoc. Relational localization requires ground-truth knowledge of spatial and structural relationships between elements. The HTML/DOM tree provides exactly this, allowing us to identify anchor neighborhoods and verify that a generated relational description uniquely identifies the target among distractors. This is substantially harder with the shallower accessibility trees typical of mobile or desktop environments.

A.2 OOD Transfer Validates Generalization

A natural concern is whether web-only training limits generalization to other platforms. Our results address this directly. Fine-tuning on Trace2Ground improves performance on ScreenSpot-Pro across all six non-web desktop domains (CAD, Creative, Dev, Office, OS, Scientific), with an average gain of +9.8 points over the base model (Tab. 2). This transfer occurs despite no desktop examples in Trace2Ground, suggesting that the supervision teaches generalizable grounding skills—handling scenarios, disambiguating identical-text candidates, and interpreting relational descriptions—rather than web-specific surface patterns.

B Data Curation Details

B.1 Human Verification

To validate dataset quality beyond automatic filtering, three expert annotators independently audited a stratified random sample of 100 test instances from each benchmark split. For every instance, annotators verified (i) instruction accuracy and (ii) bounding-box correctness. Using majority vote, we obtain 96% overall accuracy on PragmaMind (per-rater: 95%/96%/96%), 90% on RelateLoc (per-rater: 90%/90%/89%), and 86% on ContextTwin (per-rater: 85%/85%/88%). Remaining errors stem primarily from occasional visual misrecognition during data generation and residual ambiguity in the synthesized instructions.

B.2 Prompts

Figures 4–7 show the LLM prompts used to synthesize instructions for the three regimes. For **PragmaMind**, we provide two views of the same webpage: an original screenshot and a second screenshot where the target action region is highlighted with a magenta box. The model is asked to (i) describe the page and the action region, and (ii) invent

three realistic user scenarios that would naturally lead to exactly that action, while avoiding explicit mentions of UI elements or clicks. For the *type-in* variant, we additionally require each scenario to include the precise text to be typed. Outputs are constrained to a fixed JSON schema with fields for page/action descriptions and three instructions.

For **RelateLoc** and **ContextTwin**, we adopt a two-target setting: the input includes (i) a full-page screenshot with two non-overlapping boxes (magenta target, cyan competitor) and (ii) a zoomed crop revealing the boxed contents. The model first summarizes each box’s content and location, then writes an imperative instruction to localize the magenta region despite the cyan competitor. The final instruction must be less explicit than the original action description and must not reference the boxes directly, requiring compositional reasoning over content and layout. ContextTwin further encourages navigation-based localization (e.g., “after loading the page...”) instead of direct positional cues.

For the **built-in judge step**, the model receives two screenshots paired with a candidate instruction and evaluates three rejection criteria in priority order: whether the instruction expresses a clear, actionable goal; whether it leaks internal element identifiers invisible to a typical user; and whether it identifies its target solely through style-specific icon descriptions unlikely to generalize across themes. A fundamentally unactionable instruction is rejected immediately without evaluating the remaining checks. The model reasons step by step before committing to a verdict, and outputs are constrained to a JSON schema with four fields: a binary verdict, the triggered criterion (or null on acceptance), a confidence score, and a concise reason of at most 30 words.

C Dataset Statistics

This section provides additional statistics characterizing Trace2Ground splits and comparing them with existing GUI grounding datasets.

C.1 Instruction Properties

Instruction length. Fig. 8 shows the distribution of instruction lengths (in words) across datasets. Trace2Ground splits cluster substantially to the right of existing benchmarks: PragmaMind-Click, ContextTwin, and PragmaMind-Type peak between 30–40 words, RelateLoc around 20 words, while

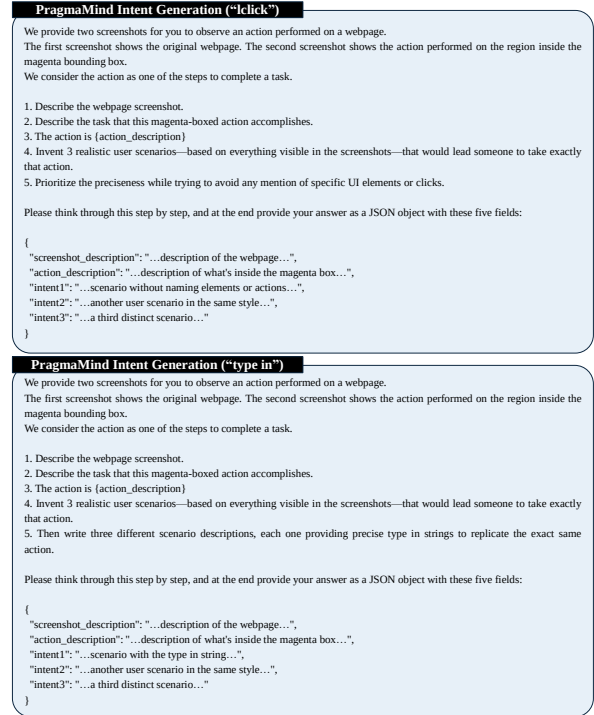


Figure 4: Prompt for PragmaMind Intent Generation.

ScreenSpot variants concentrate below 5 words and Mind2Web around 10. The longer instructions reflect the scenario-driven, context-conditioned, and relational descriptions our splits require.

Token overlap with target text. Fig. 9 shows the distribution of token-overlap scores between each instruction and its target element’s visible text, computed per sample. PragmaMind-Click and PragmaMind-Type are heavily skewed toward zero, confirming that leakage filtering suppresses verbatim matches. ContextTwin and RelateLoc show a bimodal pattern with mass near both 0 and 1: the target text often appears in the instruction (by design, since disambiguation and relational localization may reference it), but is rarely the only signal. ScreenSpot variants, by contrast, exhibit a strong peak at 1.0, indicating that the target text is reproduced almost verbatim in most instructions.

Instruction diversity. Tab. 11 reports vocabulary size, unique-instruction fraction, self-BLEU, and average length across datasets. Trace2Ground splits achieve substantially larger vocabularies (10,649–10,824) than ScreenSpot-v2 (938) and Mind2Web (3,438). Self-BLEU is moderate, reflecting templated scaffolding from the LLM synthesis (e.g., recurring phrases like “A user wants to...”), while lexical content varies substantially across samples.

RelateLoc Intent Generation (Part 1: Input)

You are provided with two images for analysis:

- Image 1: A screenshot of a webpage containing two non-overlapping bounding boxes—one magenta and one cyan. This image provides spatial and contextual information to help you locate each box's content within the full webpage layout.
- Image 2: A cropped and zoomed-in view of the area around the magenta and cyan bounding boxes. This image allows you to clearly see the visual and textual content fully enclosed within each box.

Note: The magenta and cyan bounding boxes are visual annotations only and are not part of the original webpage content.

RelateLoc Intent Generation (Part 2-1: Task Requirement)

Please examine both images carefully and complete the following tasks:

1. Content Description:
Using Image 2, describe the content fully enclosed within each bounding box (magenta and cyan). Your descriptions must be:
- Clear and concise
- Strictly limited to the content within the boundaries of each box
2. Location Description:
Using Image 1, describe the position of each box's content within the full webpage layout. Include just enough spatial or contextual information so that each box can be located.
3. Magenta Box Localization:
The action is {action_description}. We would like to localize the magenta box for this action, even in the presence of interference from the cyan box.

RelateLoc Intent Generation (Part 2-2: Task Requirement)

4. User Action Description:
For the recommended box from Task 3, provide an imperative-style user action that includes just enough information for a human user to perform it. The action must:
- Use the content description from Task 1 and the location description from Task 2, if needed
- If two boxes are adjacent, include distinguishing element attributes
- If two boxes are far apart, mention differences in their global position on the webpage (based on Image 1)
- If there are other similar elements besides the magenta and cyan boxes, consider spatial differences
- Be more ambiguous than the clear description "{action_description}", requiring some reasoning ability to correctly localize the magenta box
- Avoid referencing the boxes or annotations directly, to make the task more challenging

RelateLoc Intent Generation (Part 3: Output)

Return your response in the following JSON format:

```
{
  "magenta_box_content": "Description of the content fully enclosed within the magenta box",
  "cyan_box_content": "Description of the content fully enclosed within the cyan box",
  "magenta_box_content_location": "Location of the magenta box content within the full webpage",
  "cyan_box_content_location": "Location of the cyan box content within the full webpage",
  "action_description": "Imperative-style instruction for localizing the magenta box"
}
```

Figure 5: Prompt for RelateLoc Intent Generation.

Table 11: **Instruction diversity statistics.** Vocabulary is unique word count; Unique% is the fraction of instructions that are unique strings; Self-BLEU measures n-gram similarity across instructions (lower = more diverse); Avg. Len. is in words.

Dataset	<i>N</i>	Vocab	Unique %	Self-BLEU ↓	Avg Len
PragmaMind-Click (ours)	24,186	10,649	31.5	0.402	11.9
PragmaMind-Type (ours)	16,558	10,824	59.1	0.430	9.5
Mind2Web	1,009	3,438	100.0	0.071	14.7
ScreenSpot-v2	1,542	938	47.7	0.156	4.2

C.2 Target Properties

Target size. Fig. 10 shows target element sizes as a fraction of screen area. Trace2Ground targets are substantially smaller than those in ScreenSpot variants. Median target area is below 0.5% of the screen across all four splits, compared with ~1.5% for ScreenSpot-Desktop and Mobile. Small-target grounding is the dominant remaining failure mode in our trained models (§6).

Spatial distribution of targets. Fig. 11 shows 2D heatmaps of target element centers across datasets. Trace2Ground splits cover most of the screen, with concentrations reflecting common web layouts. PragmaMind-Type peaks near form-field

ContextTwin Intent Generation (Part 1: Input)

You are provided with two images for analysis:

- Image 1: A screenshot of a webpage containing two non-overlapping bounding boxes—one magenta and one cyan. This image provides spatial and contextual information to help you locate each box's content within the full webpage layout.
- Image 2: A cropped and zoomed-in view of the area around the magenta and cyan bounding boxes. This image allows you to clearly see the visual and textual content fully enclosed within each box.

Note: The magenta and cyan bounding boxes are visual annotations only and are not part of the original webpage content.

ContextTwin Intent Generation (Part 2-1: Task Requirement)

Please examine both images carefully and complete the following tasks:

1. Content Description:
Using Image 2, describe the content fully enclosed within each bounding box (magenta and cyan). Your descriptions must be:
- Clear and concise
- Strictly limited to the content within the boundaries of each box
2. Location Description:
Using Image 1, describe the position of each box's content within the full webpage layout. Include just enough spatial or contextual information so that each box can be located.
3. Magenta Box Localization:
The action is {action_description}. We would like to localize the magenta box for this action, even in the presence of interference from the cyan box.

ContextTwin Intent Generation (Part 2-2: Task Requirement)

4. User Action Description:
For the recommended box from Task 3, provide a user action that includes just enough information for a human user to perform it. The action must:
- Use the content description from Task 1 and the location description from Task 2, if needed
- If two boxes are adjacent, include distinguishing element attributes
- If two boxes are far apart, mention differences in their global position on the webpage (based on Image 1)
- If there are other similar elements besides the magenta and cyan boxes, consider spatial differences
- Be more ambiguous than the clear description "{action_description}", requiring some reasoning ability to correctly localize the magenta box
- If possible, create a web navigation history instead of using direct positional information. For example, use phrases like 'after loading the webpage' to refer to the top bar, or 'after navigating through the webpage' for the bottom bar. Try to rely on normal webpage navigation rather than specifying exact positions. We prefer this narrative approach, but feel free to use the imperative style to ensure accuracy.
- Avoid referencing the boxes or annotations directly, to make the task more challenging

ContextTwin Intent Generation (Part 3: Output)

Return your response in the following JSON format:

```
{
  "magenta_box_content": "Description of the content fully enclosed within the magenta box",
  "cyan_box_content": "Description of the content fully enclosed within the cyan box",
  "magenta_box_content_location": "Location of the magenta box content within the full webpage",
  "cyan_box_content_location": "Location of the cyan box content within the full webpage",
  "action_description": "Imperative-style instruction for localizing the magenta box"
}
```

Figure 6: Prompt for ContextTwin Intent Generation.

regions; ContextTwin shows clusters consistent with repeated-label patterns in navigation bars and product grids.

C.3 Action and Distractor Structure

Action type composition. Trace2Ground splits are defined around a single action type by construction (Tab. 12). PragmaMind covers both lclick and type-in actions across two splits; ContextTwin and RelateLoc focus on lclick disambiguation and relational localization.

Table 12: **Action type composition by dataset.**

Dataset	Action types
PragmaMind-Click	lclick (100%)
PragmaMind-Type	type-in (100%)
ContextTwin	lclick (100%)
RelateLoc	lclick (100%)
Mind2Web	lclick (81%), type-in (14%), select/hover/other (5%)
ScreenSpot	lclick (100%)

Distractor structure in ContextTwin and RelateLoc. Fig. 12 shows the distribution of related bounding boxes (distractors and anchors) per sample. Most samples in both regimes contain 1–3

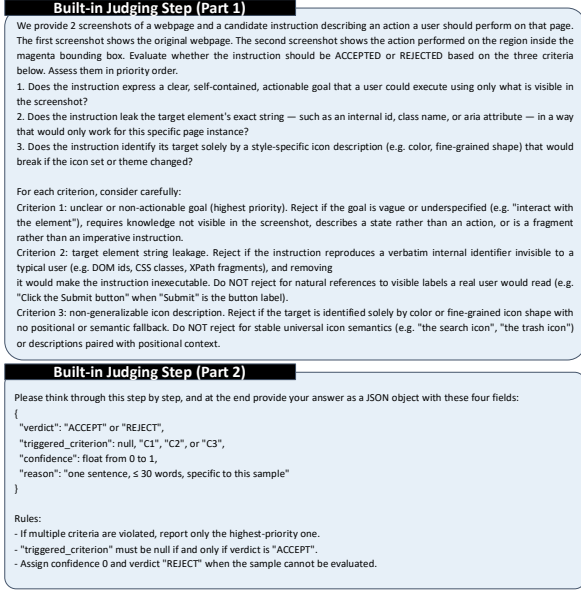


Figure 7: Prompt for built-in judge step.

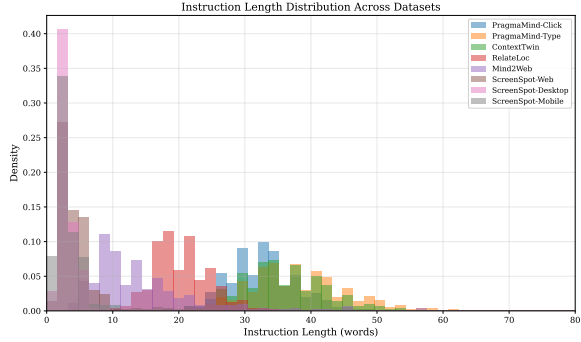


Figure 8: **Instruction length distribution across datasets.** Trace2Ground splits use substantially longer instructions than prior benchmarks, reflecting scenario-driven and relational descriptions.

designated distractors or anchors, with a long tail extending to 10+ elements for challenging layouts. The presence of at least one structured distractor per sample is enforced by construction and validated during the LLM filtering step (§3.2, §3.3).

D Supervised Fine-Tuning Details

We fine-tune the GUI grounding model ϕ (a VLM) with a standard next-token prediction objective using image, instruction text, action type, and bounding box supervision. Each training example provides

$$x = (x_{\text{img}}, q), \quad y = (a, b^*),$$

where x_{img} is the screenshot, q is the user-facing instruction, a is a discrete action type, and $b^* \in [0, 1]^4$ is the normalized target bounding box.

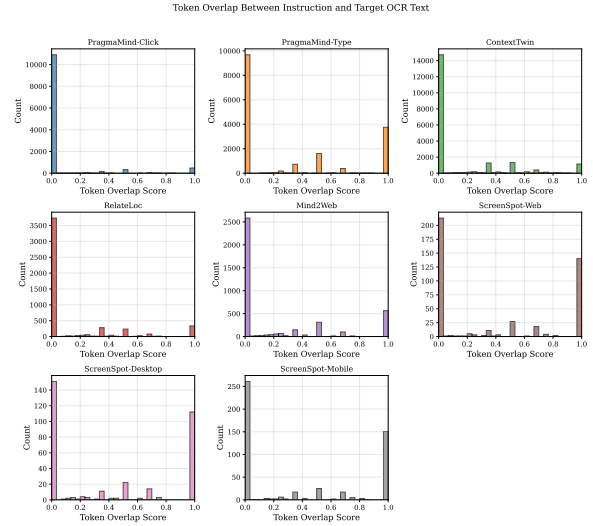


Figure 9: **Token-overlap distribution between instruction and target visible text, per dataset.** Trace2Ground splits show low or bimodal overlap; existing benchmarks concentrate at high overlap.

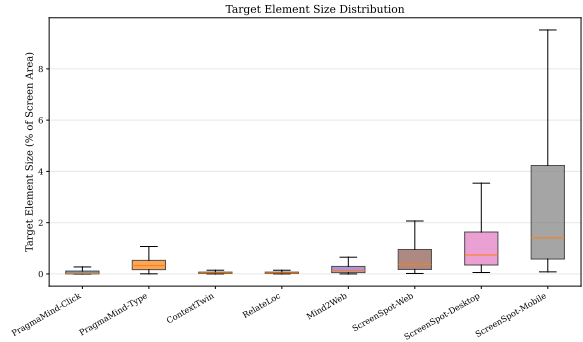


Figure 10: **Target element size as a fraction of screen area.** Trace2Ground targets are substantially smaller than those in ScreenSpot variants.

Target serialization. We serialize the target as a short textual record so ϕ can be trained purely with autoregressive language modeling. For an image of size (W, H) , we emit

$$\langle \text{act} \rangle a \quad \langle \text{box} \rangle [b_x^* W] [b_y^* H] [b_w^* W] [b_h^* H],$$

i.e., integer pixel coordinates (x, y, w, h) derived from b^* .

Objective. Let $v = f_{\text{vis}}(x_{\text{img}})$ be the visual features and let $s = (t_1, \dots, t_T)$ be the tokenized target string that linearizes (a, b^*) . We train ϕ with standard autoregressive next-token prediction under teacher forcing. Denoting by $m_t \in \{0, 1\}$ a supervision mask that is 1 on target tokens and 0

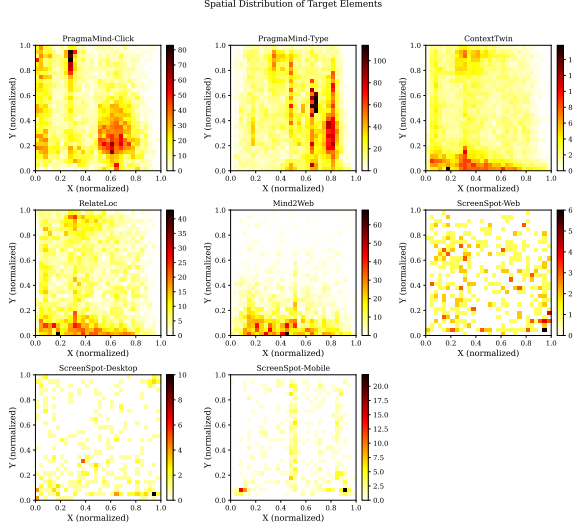


Figure 11: **Spatial distribution of target element centers.** Trace2Ground splits cover diverse screen regions with concentrations reflecting common web layouts.

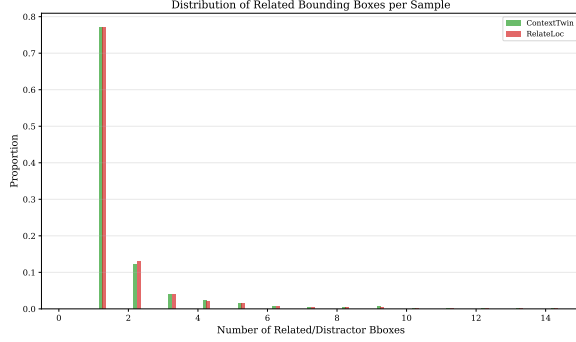


Figure 12: **Distribution of related bounding boxes (distractors and anchors) per sample in ContextTwin and RelateLoc.** Most samples contain 1–3 designated competitors; the long tail represents challenging multi-distractor layouts.

on prompt tokens, the loss is

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{\sum_{t=1}^T m_t} \sum_{t=1}^T m_t \log p_{\phi}(t_t | t_{<t}, q, v). \quad (1)$$

At inference, ϕ autoregressively generates a string that is parsed into $(\hat{a}, \hat{b})$.

Hyperparameters. We use the AdamW optimizer with learning rate 2×10^{-5} , weight decay 0.01, and train for 5 epochs. Training is conducted on a single server node with $8 \times$ NVIDIA A100 GPUs.

E Operation F1 Evaluation

In addition to Step Success Rate, we report *Operation F1* (Op. F1), which measures the accuracy

of predicted action types (lclick, type-in, etc.) independent of bounding-box localization. Let TP, FP, and FN denote true positives, false positives, and false negatives aggregated over action types. Op. F1 is the standard F1 score:

$$\text{Op. F1} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}. \quad (2)$$

Tab. 13 reports Step SR alongside Op. F1 for our backbones on the three Trace2Ground regimes. Op. F1 is near-saturated ($\geq 96\%$) for all Trace2Ground-trained models across regimes, indicating that action-type prediction is not the limiting factor for grounding performance. The main paper therefore reports Step SR as the primary metric and surfaces Op. F1 only here for completeness.

F Qualitative Examples on Trace2Ground

Figures 13–16 present representative predictions from Trace2Ground across all three reasoning splits. For **PragmaMind** (*lclick* and *type in*), the examples illustrate that successful grounding often requires combining the user-driven scenario intent. For **RelateLoc**, we highlight cases with strong distractors where correct localization depends on relational reasoning (e.g., selecting the target element relative to nearby panels or matched content). **ContextTwin** showcases the most ambiguous settings, where multiple regions share similar surface appearance and the model must resolve the target through subtle contextual or narrative cues. Overall, the qualitative results align with our quantitative findings: Trace2Ground predictions (orange) closely match ground-truth regions (blue) in most cases, indicating improved reasoning capabilities.

G Qualitative examples on ScreenSpot-Pro

Models trained on Trace2Ground shows remarkable capabilities on out-of-distribution benchmarks. Even though the model is trained on web data, it shows adaptation on ScreenSpot-Pro (Li et al., 2025b) Cad in Figure 17, Creative in Figure 18, Dev in Figure 19, Office in Figure 20, OS in Figure 21, and Scientific in Figure 22.

H Broader Impact

More reliable GUI grounding has potential benefits in accessibility, assistive automation, and software testing, where errors on small or context-dependent UI elements directly affect end users. As with any

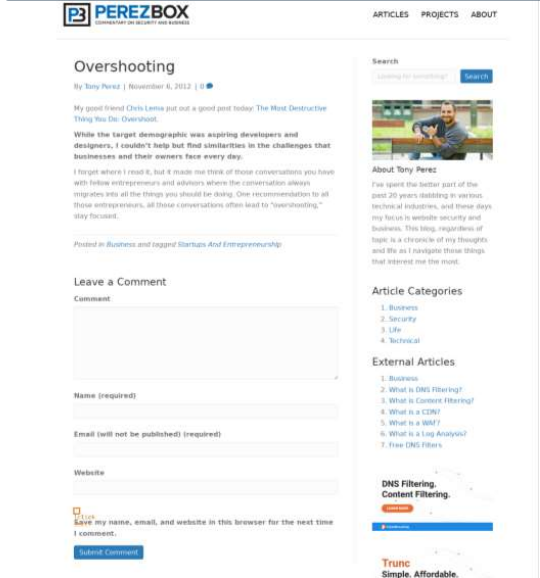
Table 13: **Full Step SR and Operation F1 results on Trace2Ground diagnostic regimes (%)**. Qwen2.5-VL* denotes the public-data SFT baseline.

Model	PragmaMind-Click		PragmaMind-Type		ContextTwin		RelateLoc	
	Step SR	Op. F1	Step SR	Op. F1	Step SR	Op. F1	Step SR	Op. F1
Qwen2.5-VL* 3B	46.39	99.18	14.15	41.81	35.96	99.91	39.92	99.96
Trace2Ground 3B	94.29	99.98	90.54	96.95	76.61	99.97	72.78	99.98
Qwen2.5-VL* 7B	48.39	99.71	30.96	54.32	45.34	99.92	45.42	99.98
Trace2Ground 7B	94.17	99.98	93.42	98.16	77.15	99.97	74.40	100.00

automation technology, however, improved grounding can also be misused, for example to automate scraping, account creation, or other unauthorized interactions at scale. We recommend that practitioners deploying models trained on Trace2Ground implement appropriate safety checks, user consent mechanisms, and monitoring, and that they consider accessibility goals when designing systems that rely on automated UI interaction.

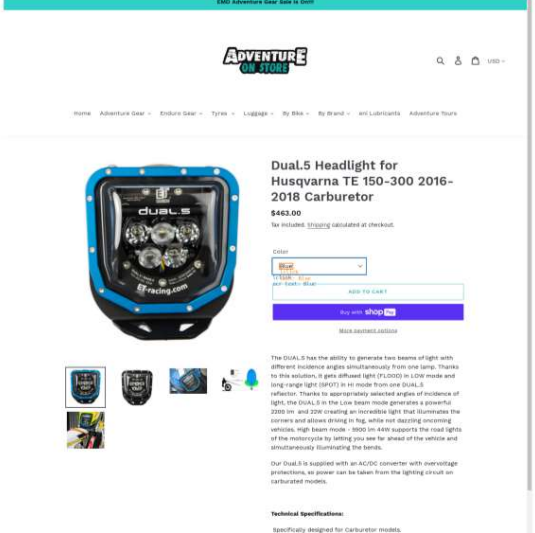
PragmaMind “Iclick”

Intent: A regular reader of the PerezBox blog who frequently engages with posts about security and business topics wants to streamline their commenting experience by saving their personal details.



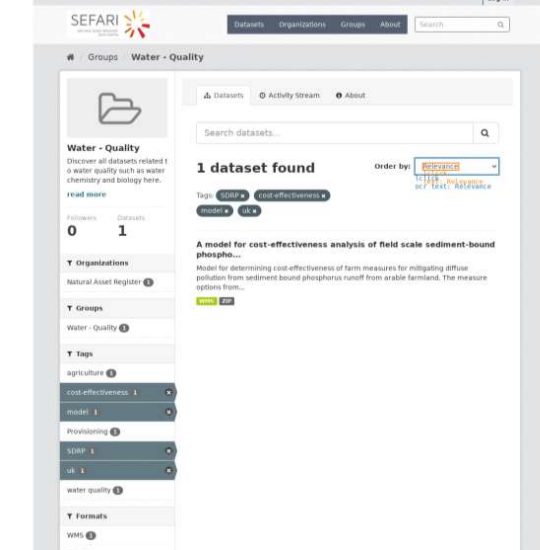
PragmaMind “Iclick”

Intent: A customer wants to check if the headlight comes in different color options besides the blue trim that's shown in the product images before making their purchase decision.



PragmaMind “Iclick”

Intent: A researcher wants to change how the water quality datasets are sorted because they're looking for the most recently added models rather than the most relevant ones to get the latest research on phosphorus pollution mitigation.



PragmaMind “Iclick”

Intent: A job seeker wants to narrow down their search results to only view positions in a specific field like underwriting or IT, as they have expertise in that particular area.

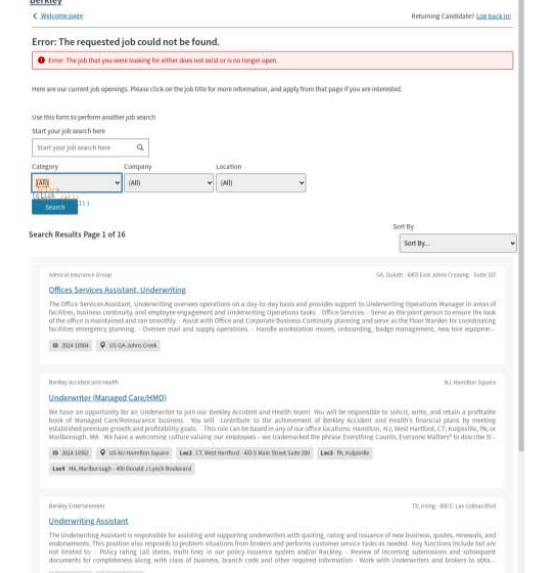
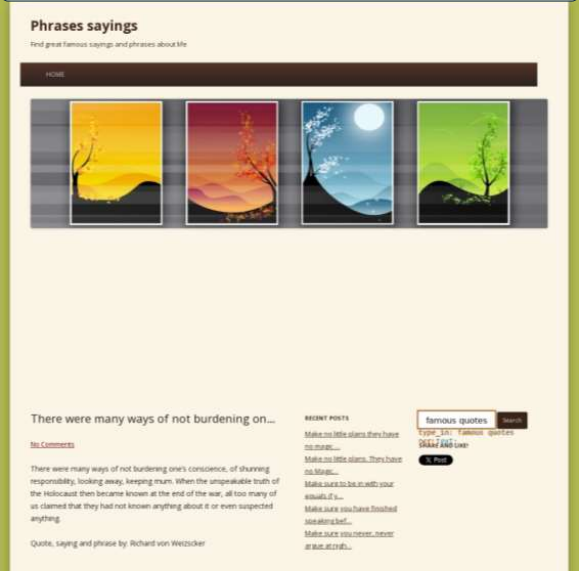


Figure 13: Qualitative examples on PragmaMind 'Iclick' split. Blue denotes groundtruth and orange denotes the prediction.

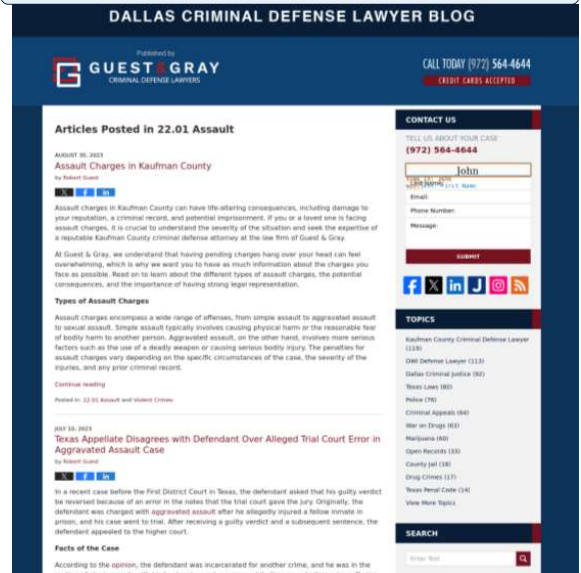
PragmaMind “type in”

Intent: I'm reading this blog about Holocaust denial quotes and remembered a famous quote about responsibility I once read. I'll search for 'famous quotes' to see if this site has a collection of other thought-provoking sayings on similar themes.



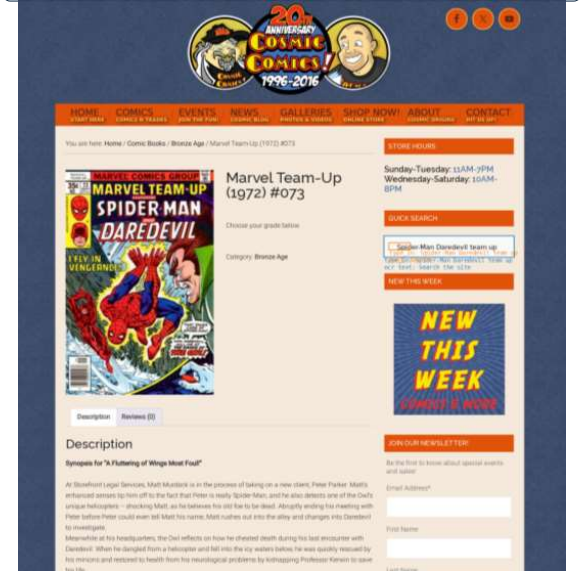
PragmaMind “type in”

Intent: John is facing assault charges in Kaufman County and wants to contact Guest Gray Law Firm for legal representation, so he begins filling out the contact form with his first name.



PragmaMind “type in”

Intent: I'm a collector looking for classic team-up issues between Spider-Man and Daredevil, so I'll search 'Soldier-Man Daredevil team up' to find all available issues in the store's inventory.



PragmaMind “type in”

Intent: John wants to stay informed about Connecticut House Republicans' positions on the gas-powered vehicle ban mentioned in the banner, so he partially fills out the newsletter form with his first name 'John' and email address 'john.smith@email.com', planning to submit the minimal required information.



Figure 14: Qualitative examples on PragmaMind 'type in' split. Blue denotes groundtruth and orange denotes the prediction.

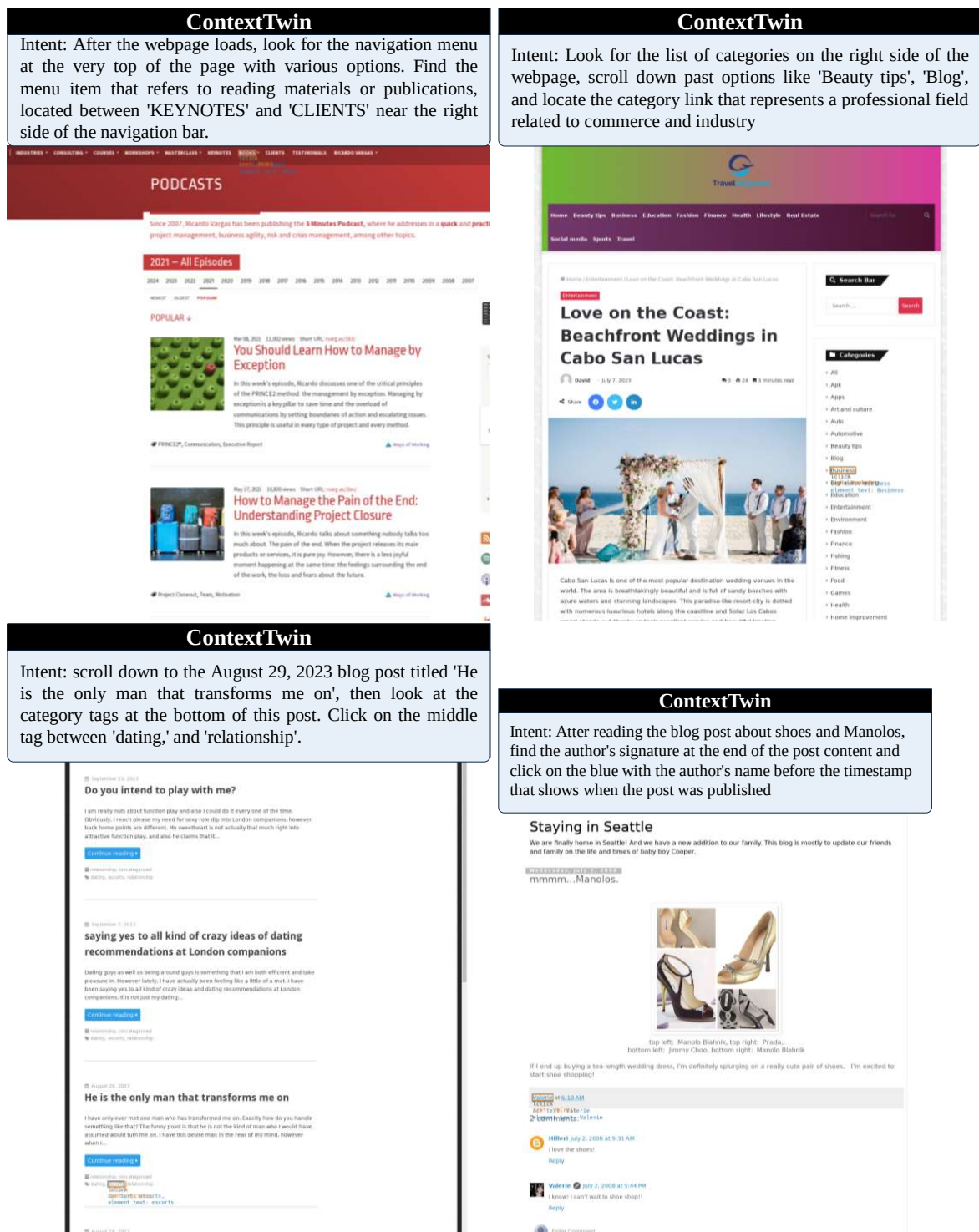


Figure 16: Qualitative examples on ContextTwin split. Blue denotes groundtruth and orange denotes the prediction.

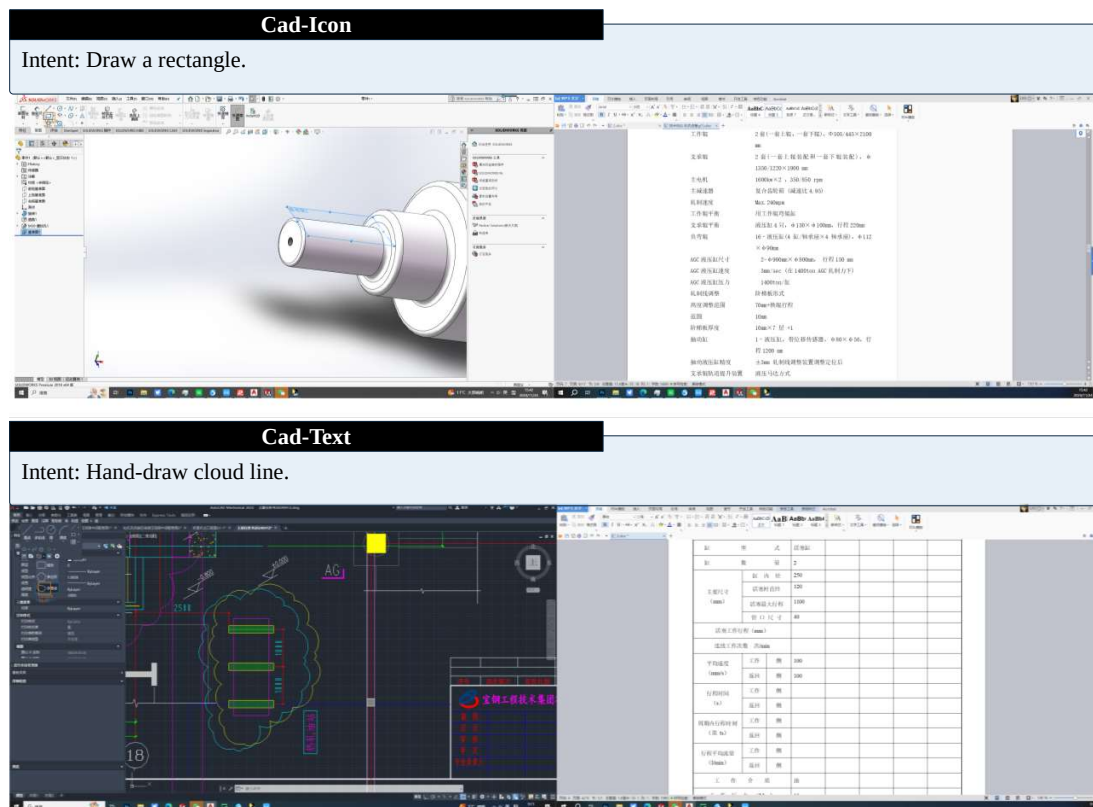
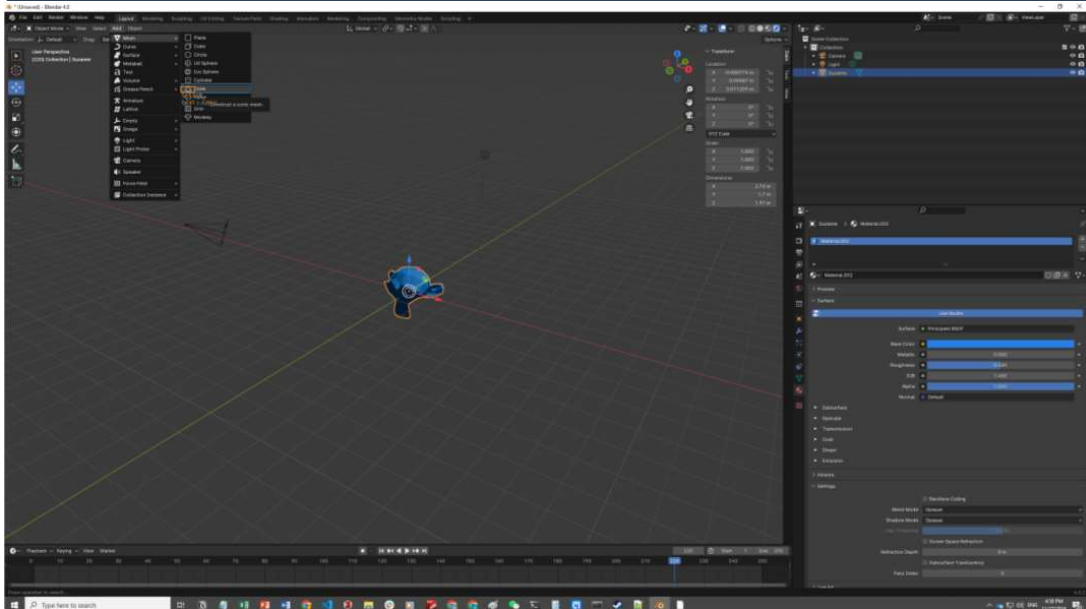


Figure 17: Qualitative examples from ScreenSpot Pro Cad Icon and Text.

Creative-Icon

Intent: Add a new cone object to decorate the monkey.



Creative-Text

Intent: Enter sculpt mode to use Inflate brush to enlarge some specific areas.

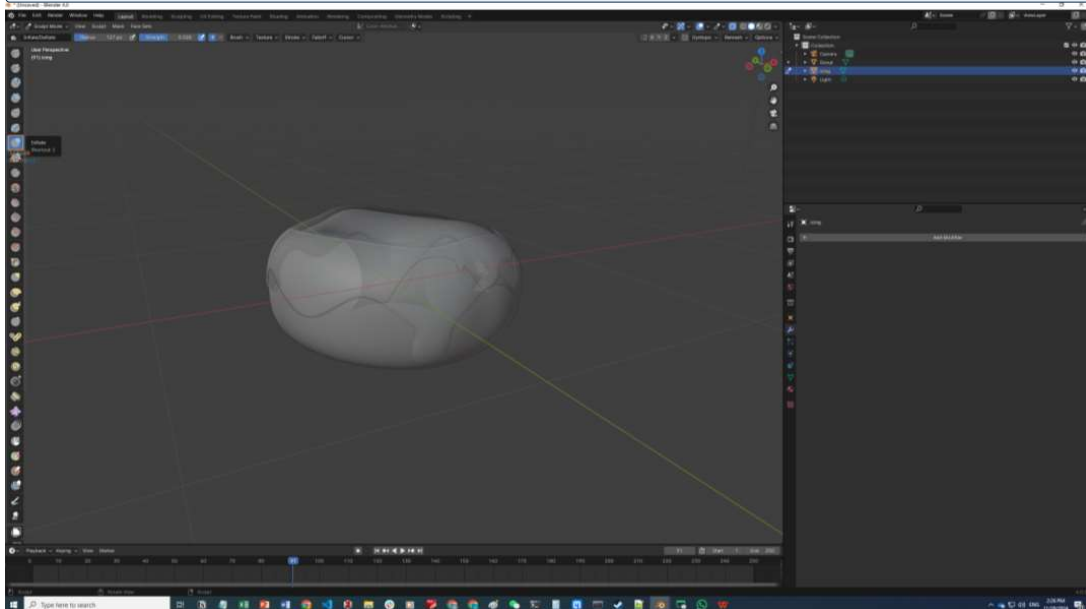
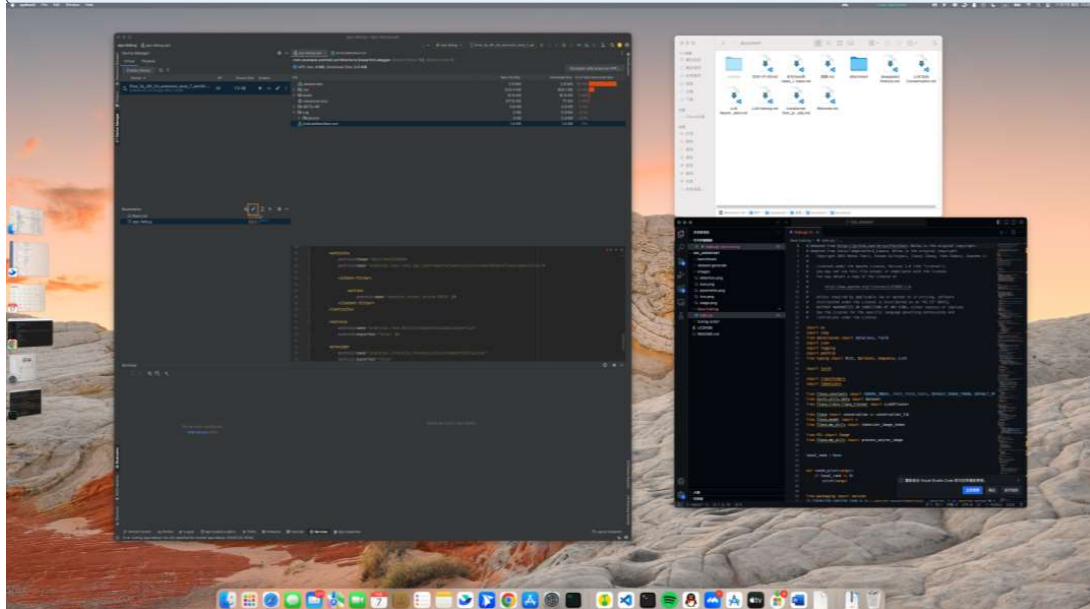


Figure 18: Qualitative examples from ScreenSpot Pro Creative Icon and Text.

Dev-Icon
Intent: Rename the bookmark name in android studio.



Dev-Text
Intent: Open marscode ai logger in android studio.

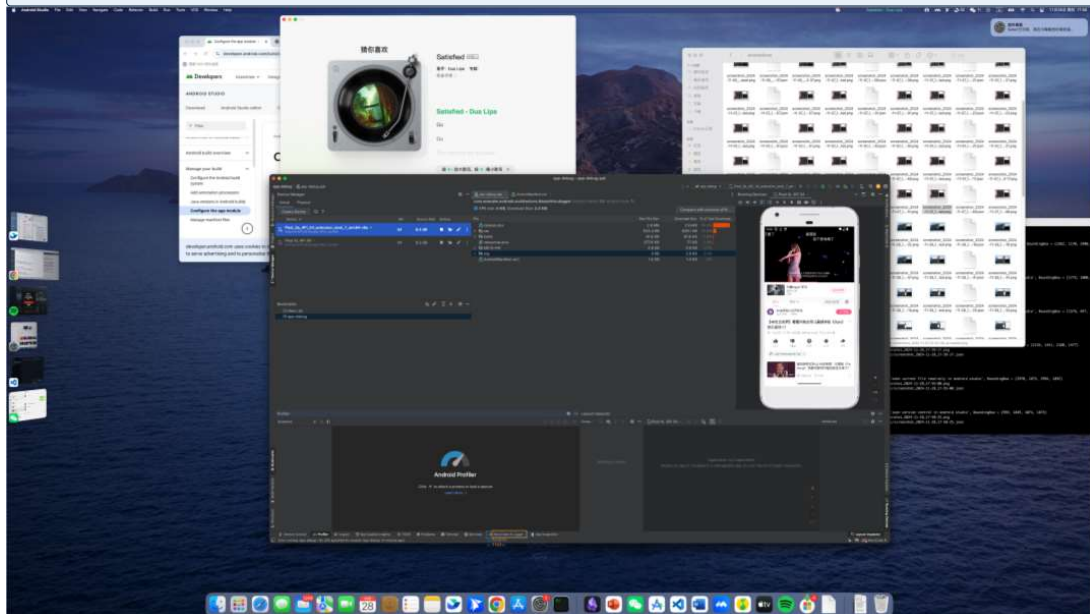
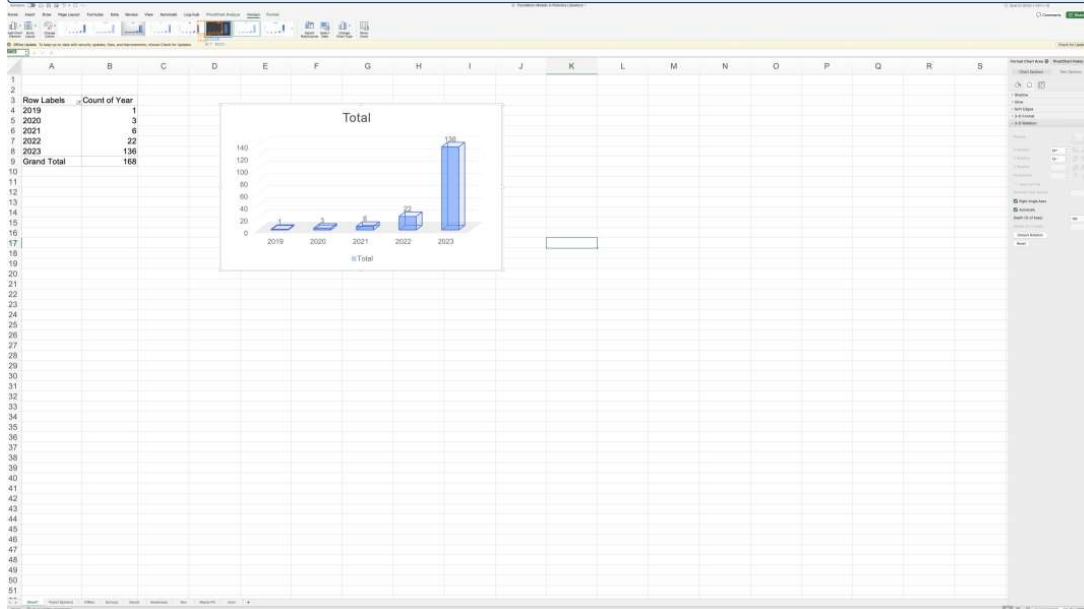


Figure 19: Qualitative examples from ScreenSpot Pro Dev Icon and Text.

Office-Icon

Intent: Set to the design with black background.



Office-Text

Intent: Confirm turning off.

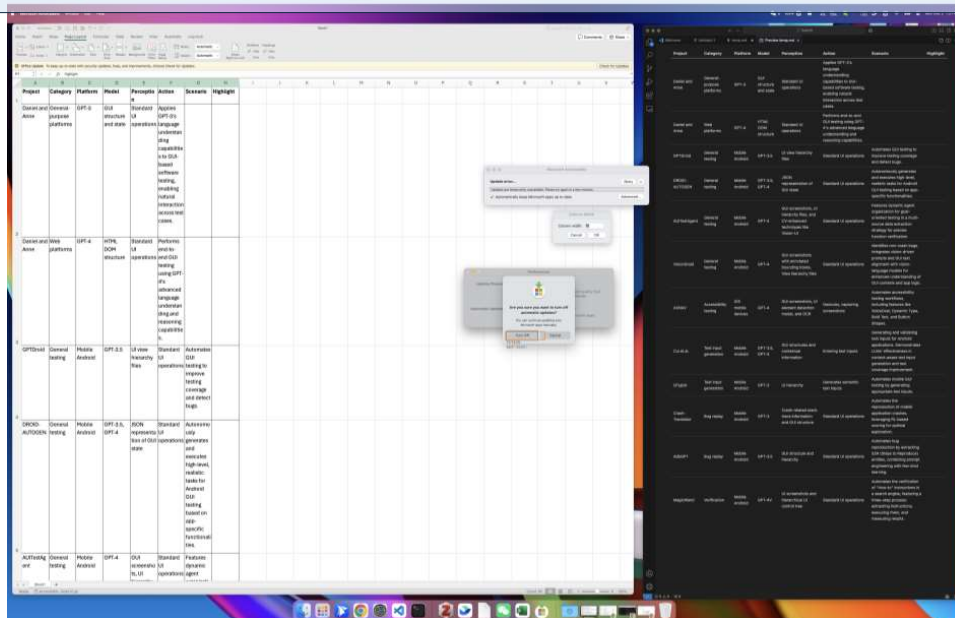


Figure 20: Qualitative examples from ScreenSpot Pro Office Icon and Text.

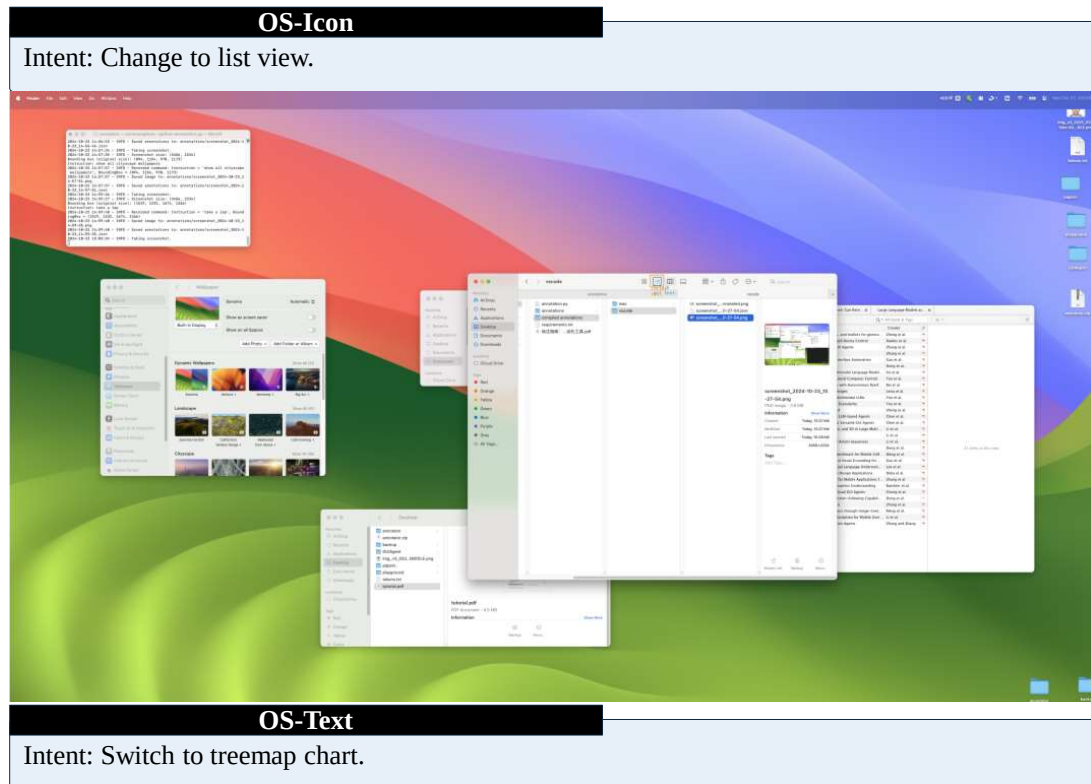
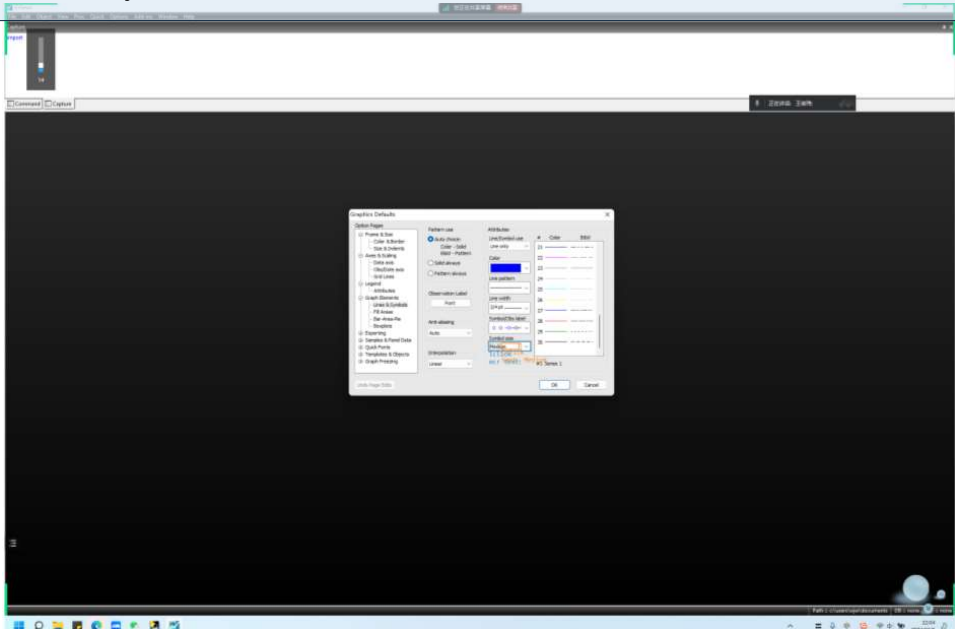


Figure 21: Qualitative examples from ScreenSpot Pro OS Icon and Text.

Scientific-Icon

Intent: Choose symbol size.



Scientific-Text

Intent: Input maximum errors before halting.

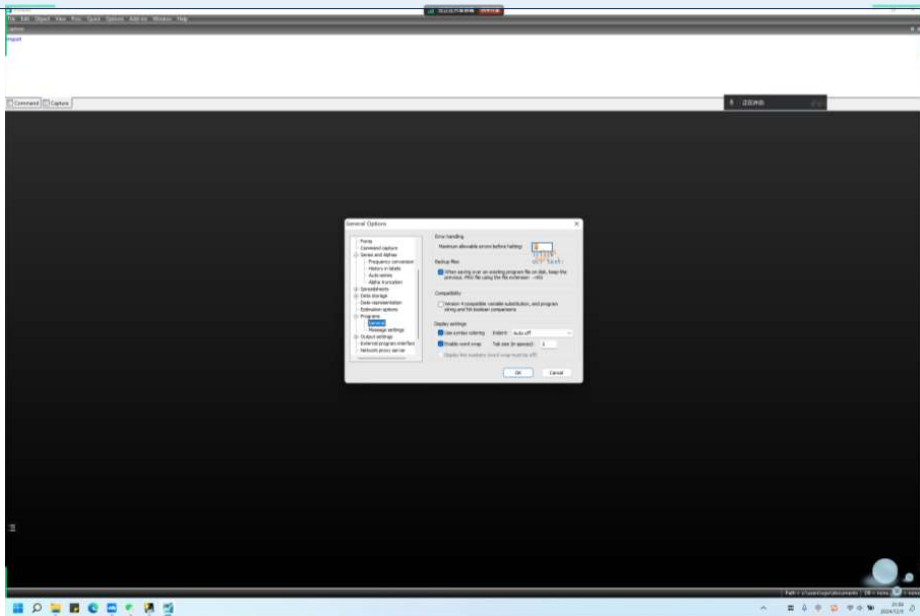


Figure 22: Qualitative examples from ScreenSpot Pro Scientific Icon and Text.