AgentTrek: Agent Trajectory Synthesis via Guiding Replay with Web Tutorials

Yiheng Xu*♠ Dunjie Lu*♠ Zhennan Shen*♠ Junli Wang♠ Zekun Wang♠
Yuchen Mao♠ Caiming Xiong♣ Tao Yu♠
♠University of Hong Kong ♣Salesforce Research
♠{yhxu,tyu}@cs.hku.hk ♣cxiong@salesforce.com
https://agenttrek.github.io

Abstract

Graphical User Interface (GUI) agents hold great potential for automating complex tasks across diverse digital environments, from web applications to desktop software. However, the development of such agents is hindered by the lack of high-quality, multi-step trajectory data required for effective training. Existing approaches rely on expensive and labor-intensive human annotation, making them unsustainable at scale. To address this challenge, we propose AgentTrek, a scalable data synthesis pipeline that generates high-quality web agent trajectories by leveraging web tutorials. Our method automatically gathers tutorial-like texts from the internet, transforms them into task goals with step-by-step instructions, and employs a visual-language model (VLM) agent to simulate their execution in a real digital environment. A VLM-based evaluator ensures the correctness of the generated trajectories. We demonstrate that training GUI agents with these synthesized trajectories significantly improves their grounding and planning performance over the current models. Moreover, our approach is more cost-efficient compared to traditional human annotation methods. This work underscores the potential of guided replay with web tutorials as a viable strategy for large-scale GUI agent training, paving the way for more capable and autonomous digital agents.

1 Introduction

Graphical User Interfaces (GUIs) are a fundamental medium for human-computer interaction, enabling users to perform tasks across various digital platforms. Automating GUI operations through agentic automation has the potential to significantly enhance productivity by enabling autonomous task completion using human-centric tools. Additionally, this approach can foster the development of advanced AI systems capable of learning from rich digital environments.

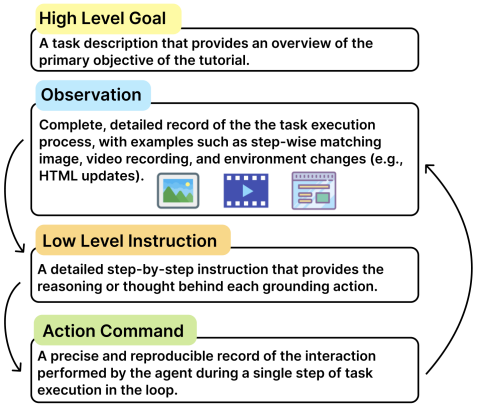

Figure 1: Expected Agent Trajectories

Recent advancements in large language models (LLMs) have endowed the models with powerful abilities in understanding, reasoning, and decision-making, which are essential for the evolution of GUI agents in diverse contexts such as web (Zheng et al., 2024), desktop (Xie et al., 2024), and mobile applications (Zhang et al., 2023). Despite these advancements, the performance of GUI agents remains suboptimal. Contemporary Large Language Models (LLMs) are primarily engineered and trained on datasets optimized for generating informative responses (Ouyang et al., 2022; OpenAI, 2024). Their architecture and training paradigms are

*Equal contribution

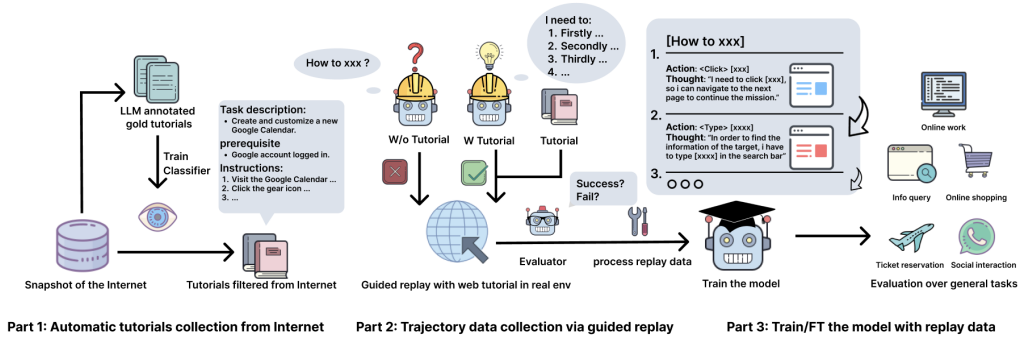

Figure 2: **Overview of the `AgentTrek` Pipeline**: (1) *Automatic Tutorial Collection from the Internet:* Tutorial-related data is extracted and filtered from internet sources using heuristic methods and a FastText model. An LLM processes the filtered textual data, transforming it into structured tutorials. (2) *Trajectory data collection via guided replay:* A VLM agent interacts with the real digital environment guided by tutorials, while high-quality trajectory data, including observations, actions, and reasoning, is collected. Another VLM evaluator acts as a judger to further improve the effectiveness of the synthetic dataset. (3) *Training and fine-tuning with replay data:* The collected trajectory data is used to train and fine-tune GUI agent models, which are evaluated on standard agent benchmarks, demonstrating significant improvements.

not inherently designed to make complex, sequential action decisions that require long-term observation and historical context. Consequently, training GUI agents with multi-step trajectory data is crucial to improving their capabilities.

High-quality agent trajectories contain several key components: a high-level goal, a sequence of interleaved observations, natural language reasoning, and grounded actions (as shown in Figure 1). Unfortunately, such data is not readily available on the internet like textual or image data, as it involves complex situational reasoning and multimodal interactivity. Existing approaches typically rely on human annotation to collect these trajectories (Deng et al., 2024; Rawles et al., 2023; Li et al., 2024), a process that is both expensive and not scalable.

To address this data scarcity, data synthesis has emerged as a vital approach in AI system development. Synthesizing agent trajectories presents significant challenges due to the need for interwoven natural language instructions, visual observations, and context-specific actions that must be accurately grounded in the GUI environment. Although there have been some successful applications of LLMs in data synthesis pipelines (Ye et al., 2022; Peng et al., 2023; Qin et al., 2023), these complexities still make GUI trajectory synthesis particularly demanding.

In this work, we present `AgentTrek`, a scalable data synthesis pipeline specifically designed for training GUI agents. We begin by automatically gathering and filtering tutorial-like text from the web, which describes GUI tasks and workflows in web environments. These tutorials are then transformed into agent tasks with high-level objectives and detailed step-by-step instructions. Using a visual-language model (VLM) agent, we simulate the execution of these tasks, guided by the synthesized tutorials. An evaluator model is also employed to subsequently verify whether the goal was successfully achieved. Through this comprehensive pipeline, we efficiently generated a large volume of high-quality web agent trajectories.

Our experimental results demonstrate that training GUI agent models with these synthesized web agent trajectories not only improves their performance but also enables them to surpass the capabilities of their initial teacher models, which is the replay model GPT-4 in our case. Compared to traditional human-annotated data pipelines, our method is significantly more cost-effective, emphasizing the scalability and economic viability of the `AgentTrek` pipeline.

- We introduce `AgentTrek`, a novel pipeline that leverages web tutorials to synthesize high-quality web agent trajectory data at scale, effectively bridging the gap between LLM capabilities and the demanding need for multi-step, context-rich training data for GUI agents.

- Extensive experiments demonstrate that agents trained with our synthesized data outperform those trained on existing datasets in both grounding and planning capabilities, validating the effectiveness of `AgentTrek`.
- Our pipeline significantly reduces the cost and scalability barriers of human-annotated data collection, providing a practical approach for large-scale GUI agent training through data synthesis.

Table 1: **Comparison of AgentTrek with other trajectory datasets for training.** For the calculation of dataset size and average steps, see Appendix A.

Datasets	Size	Average Steps	HTML	AxTree	Intermediate Reasoning	Video	Matching Screenshot	Website	Task Inst. Level
RUSS	80	5.4	Yes	No	No	No	No	22	Low
ScreenAgent	203	4.3	No	No	Yes	No	Yes	-	High & Low
WebLINX	969	18.8	Yes	No	No	No	Yes	155	High & Low
MM-Mind2Web	1009	7.3	Yes	No	No	No	No	137	High
GUIAct	2482	6.7	No	No	No	No	Yes	121	High
AgentTrek (Ours)	**10398**	**12.1**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	127	**High & Low**

2 METHOD

We introduce a pipeline to collect and process GUI tutorials from the internet for training VLMs/LLMs in web automation tasks. The method comprises three main steps:

1. **Collecting Tutorials**: We extract web interaction tutorials from large datasets using keyword filtering and language models to identify and standardize relevant content.
2. **Guided Replay**: An agent uses these tutorials to perform tasks in a web environment, interacting with real websites while we record its actions and thoughts.
3. **Model Training**: We train a visual agent model that relies on screenshots and standard GUI actions, enhancing its web navigation capabilities with the collected data.

This approach enables efficient training of VLMs without extensive manual annotation, offering a scalable solution for automating web tasks.

2.1 AUTOMATIC TUTORIALS COLLECTION FROM INTERNET

We extract web interaction tutorials from the RedPajama dataset (Computer, 2023) and apply a rule-based filter to create a preliminary dataset. A subset is annotated by an advanced LLM to generate labeled samples for training a FastText classification model (Joulin et al., 2017), the tutorial classifier. This classifier improves data quality through further filtering. Finally, LLMs tag and paraphrase the tutorials into a standardized format for the replay phase in Section 2.2.

2.1.1 PREFILTER FUNCTION

Although GUI tutorials are widespread online, they represent a small fraction of web content, requiring a pre-filter to identify relevant material. Tutorials often share common keywords like "click", "type", and platform terms such as "macOS" and "Windows". We created a rule-based filter using keyword lists from official sources. Leveraging RedPajama's 20+ billion URLs, the filter uses **Keyword Matching**, evaluates **Length**, and filters by **URL Format** for relevance.

Validated using 180 positive and 105 negative ground-truth samples, the prefilter achieved a 92.69% recall rate on positive samples, ensuring both diversity and quantity. After filtering, the dataset size is reduced from 20.8 billion to 68.8 million entries (Figure 4).

2.1.2 LLM LABELER

Initial rule-based filtering reduces the context, but true positive tutorials remain scarce. To improve quality, we use the advanced LLM GPT-4O MINI for automated labeling, given its ability to analyze

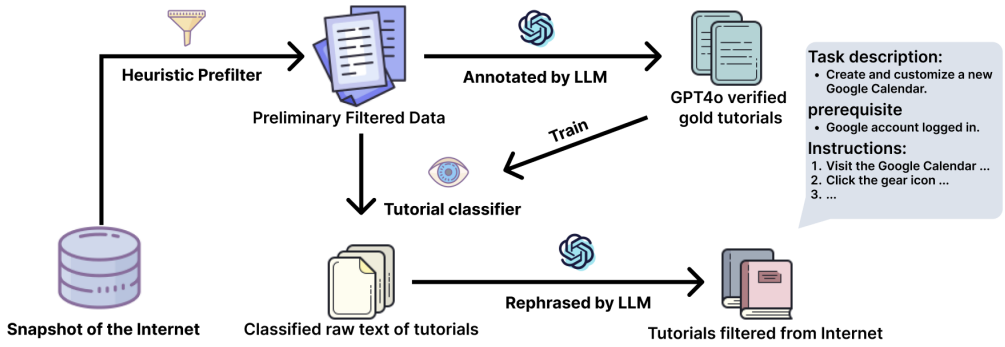

Figure 3: **Overview of the tutorial filtering and classification pipeline.** Starting with Redpajama, the data is prefiltered, annotated by an advanced LLM, and used to train a tutorial classifier. The classifier further filters the raw text, which is then paraphrased into structured tutorials with task descriptions, prerequisites, and step-by-step instructions.

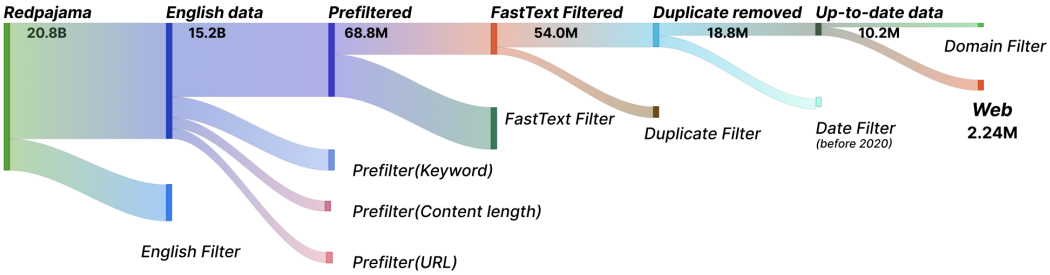

Figure 4: The data flow during the early stages of our pipeline.

complex text. We tested it on a manually annotated validation set, where it achieved an F1 score of nearly 90%. In cases of disagreement between human and LLM annotations, the LLM identified tutorial content in long texts that humans missed. This suggests that GPT-4O MINI may outperform humans in webpage labeling, helping efficiently generate a large labeled dataset for the next phase.

2.1.3 FASTTEXT FILTER

After automated labeling, we used FastText, an n-gram-based model, to classify tutorial text as either tutorial or non-tutorial, providing a binary output with a confidence score for better accuracy. We trained the model using a dataset of around 90,000 samples, combining LLM-labeled and human-labeled data, with a 95:5 train-test split. The model showed strong performance. Using this classifier, we curated the initial filtered dataset, gathering about 18.8 million tutorial-like web text samples.

Table 2: Performance of Filters.

Metric	Precision	Recall	F1
Prefilter	0.69	0.61	0.60
LLM	0.885	0.885	0.89
FastText	0.895	0.895	0.89

2.1.4 TAG & PARAPHRASE

After filtering with FastText, we tag and paraphrase the tutorial content for further processing, including extracting meta-info and standardizing the format. To manage length and noise, we use GPT-4O MINI for efficient tagging and paraphrasing, ensuring alignment with the template and gold-standard examples. The template includes: **Platform and Target** (e.g., macOS, Windows),

Task Description, **Prerequisites**, **Step-by-Step Instructions**, and **Expected Outcome**. Tagging and paraphrasing 1,000 entries costs about 0.89 dollars.

2.2 TRAJECTORY DATA COLLECTION VIA GUIDED REPLAY

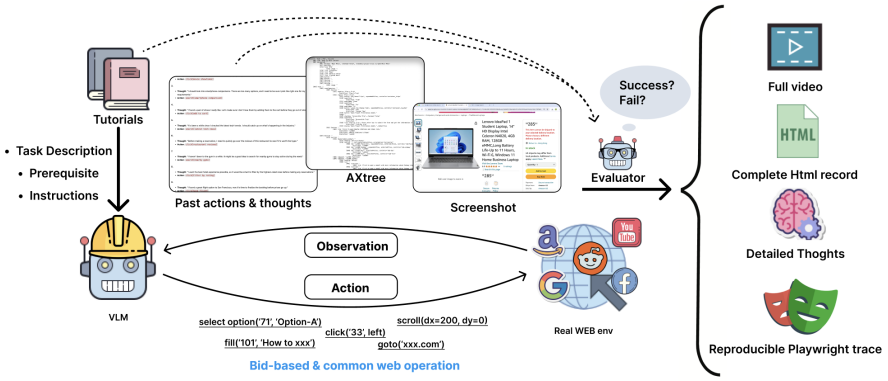

Figure 5: **Overview of Guided Replay data collection and evaluation pipeline.** A VLM agent is provided with the filtered and formatted tutorials, then observes and interacts with the real environment during execution, while all the actions and intermediate thoughts are recorded as data trajectory. The final result is evaluated by an advanced VLM to ensure the correctness.

2.2.1 TRAJECTORY DATA DEFINITION

The trajectory data generated by our pipeline enhances an agent's web navigation by integrating high-level planning, low-level instructions, and grounded operations. Each data instance includes the following components:

Task Information. Task metadata such as platform, description, prerequisites, instructions, and expected outcomes, supporting both planning and execution.

Screenshots and Video Recordings. Visual documentation that captures every step of the process, providing a comprehensive overview of the agent's trajectories.

Reproducible Native Trace. Captured via Playwright, including DOM snapshots, HTML, network flow, and action sequences for full reconstruction and detailed analysis.

Post-processed Textual Trajectory. it includes *Task Metadata*, summarizing the task for decision-making, *Observations*, providing context, *Intermediate Reasoning*, offering insights into the agent's decision process, and *Action Sequence*, capturing web interaction details for fine-tuning.

2.2.2 GUIDED REPLAY WITH TUTORIALS

Although we've collected high-quality tutorials, there is still a gap in the trajectory data needed to train a better agent model. To fill this gap, we use BrowserGym (Drouin et al., 2024) to allow the model to replay tasks based on the generated tutorials.

BrowserGym is a flexible environment for web task automation in Chromium, allowing Visual Language Model (VLM) agents to perform web-based operations (Drouin et al., 2024). Agents are provided with tagged tutorials and a *target_web_url*, guiding them through tasks with step-by-step instructions and expected outcomes.

The agent's initial observations include a screenshot of the webpage's viewport and the accessibility tree (AXTree), excluding the HTML file due to its size and irrelevance to visual agents. Actions are performed using Playwright functions like *click*, *select_option*, and *clear*, while detailed traces, including target elements, coordinates, screenshots, and DOM snapshots, are recorded along with the agent's internal thoughts.

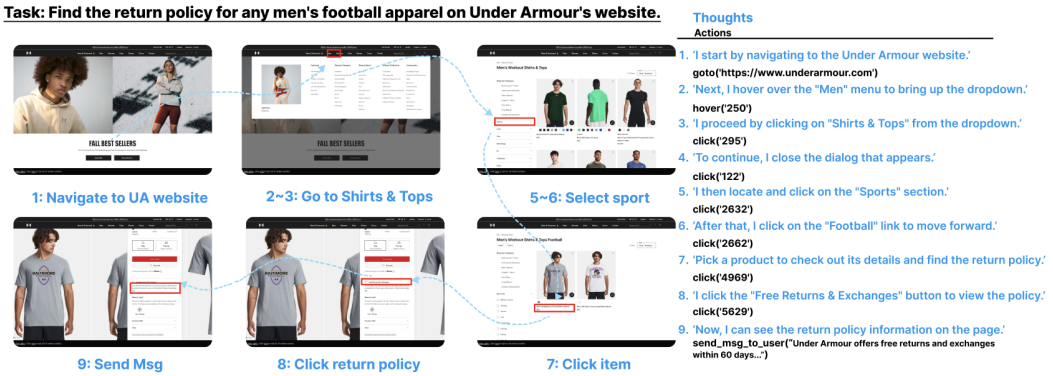

Figure 6: **Guided replay example.** This example demonstrates an agent's execution of finding the return policy for men's football apparel, showcasing its actions alongside the corresponding inner thoughts.

Token usage is about 8,027 per step and 86,114 per task. Replaying 1,000 tasks with GPT-4o-08-06 costs around 215 dollars. Cost details are in C.

2.2.3 Evaluation of Trajectory

While a large amount of guided replay data has been recorded, it's essential to extract the segments that truly enhance the agent's performance. Recent work by (Pan et al., 2024) shows that Visual Language Models (VLMs) can evaluate trajectory data using recorded images and interactions. VLMs are scalable, cost-effective, and provide transparent evaluations. Thus, we implemented a VLM Evaluator to improve data quality.

VLM Evaluator Design. To ensure trajectory data quality, we define *effectiveness* by two factors: adherence to task instructions and completion of key components. We use GPT-4o as the core of our VLM evaluator, which assesses recorded trajectories through a structured prompt. The evaluator receives the task description d, the agent's action history $\overline{a}$, and inner thoughts $\overline{r}$. The sequence follows this format: {task description; inner thought 1; action 1; inner thought 2; action 2; ...}, as shown in Figure 5. The VLM provides a trajectory-level assessment, analyzes each step, and identifies the earliest point of failure.

Table 3: Evaluator Accuracy Comparison

Trajectory	Evaluator	Acc.
Replayed Web Tutorials	GPT-4o	84.0%
WebArena Results	GPT-4V	80.6%
	Cap. + GPT-4	82.1%
	Cap. + Mixtral	74.4%

Table 4: Cost Breakdown

Phase	Cost/1k ($)	Model
T&P	0.89	`gpt-4o-mini`
Replay	215.36	`gpt-4o`
Eval	3.10	`gpt-4o`
Total	**219.35**	–

Validation on Human-Annotated Set. While the capabilities of Vision Language Models are well-known, validation is necessary. To assess the evaluator's performance, we manually reviewed 1,081 trajectories and created a validation set of 558 samples with human-annotated justifications.

As shown in Table 3, despite handling various input formats, the evaluator achieved strong performance. According to observations in Appendix D, the evaluator often applies stricter standards than human reviewers, demonstrating its robustness in identifying effective trajectories.

2.3 Training and Fine-tuning with Trajectory Data

We chose a visual agent model that uses screenshot-based observations instead of accessibility trees or textual representations for three main reasons. First, GUIs are visual, and mapping instructions

to visuals aligns better with human cognition. Second, textual representations like HTML can be verbose and computationally expensive. Lastly, image-based inputs help unify observations across different websites, improving generalization.

2.3.1 VISION-BASED WEB AGENT

For the vision-based agent, we use pure vision as input, eliminating the need for UI source code understanding. This reduces token consumption, with high-resolution models like Qwen2-VL using only 1,200 tokens for a 720p image, compared to 4,000 tokens for HTML. The action space is based on pyautogui commands, which interact with visual UI elements. We map playwright actions to pyautogui commands and use a pluggable action system for specific tasks like select_option.

2.3.2 TEXT-BASED WEB AGENT

The text-based agent utilizes the accessibility tree as its primary observation source, enabling it to understand and interact with web elements based on their semantic structure. Actions are performed using playwright commands, which allow the agent to manipulate web elements through the accessibility tree. This approach is well-suited for tasks that require a deeper understanding of web structures, such as forms, links, and other interactive elements.

2.3.3 MODEL ARCHITECTURE AND TRAINING

For vision-based agents, we use Qwen2-VL (Wang et al., 2024) with NaViT as an image encoder, enabling dynamic resolution support Dehghani et al. (2023). This approach allows the model to process images efficiently, making it ideal for GUI tasks by mapping intents directly to visual inputs. We fine-tune the model using 10,000 trajectories from the AgentTrek dataset to enhance grounding and planning capabilities.

For the text-based agent, we fine-tune Qwen2 LLMs of various sizes, including 7B and 32B, using 6,000 agent trajectories from the AgentTrek dataset. These trajectories consist of accessibility tree observations and actions using playwright commands, enabling the agent to understand and interact with web elements through their semantic structure. This fine-tuning improves the agent's ability to interpret and respond to textual cues, enhancing its task planning and interaction with web content.

3 EXPERIMENTS

AgentTrek autonomously collects thousands of trajectories with multimodal data, including screenshots, accessibility trees, reasoning, and detailed actions. This data is ideal for fine-tuning LLMs for text-based agents and VLMs for visual web agents. We demonstrate its utility by training both text-based and vision-only agents to perform assigned tasks.

3.1 EXPERIMENTAL SETUP

Evaluation For Text-based Web Agent. To demonstrate the capability of the text-based agent, we select WebArena Zhou et al. (2023) as evaluation benchmark. WebArena, based on real websites, creates multiple virtual environments and uses various evaluation methods to assess the task completion rate, making it more suitable for real-world task completion evaluation.

Evaluation For Vision-based Web Agent. To validate our dataset's effectiveness, we evaluate its impact on improving the model's grounding and planning capabilities using two benchmarks. (1) ScreenSpot (Cheng et al., 2024), a GUI visual grounding benchmark with 1.2K single-step instructions and target element bounding boxes, covering mobile, desktop, and web environments. We focus on web-based performance since our data is from the web. (2) Multimodal-Mind2Web (Deng et al., 2024), an extension of the Mind2Web benchmark (Deng et al., 2024), which includes cross-task, cross-website, and cross-domain categories, ranked by their deviation from the training data.

3.2 MAIN RESULTS

WebArena. The experimental results from table 5 show that: (1) AgentTrek's textual trajectories significantly boost performance on WebArena, surpassing open-source baselines and GPT-4o. (2) Considering that WebArena is an OOD web agent benchmark featuring self-hosted websites, strong results on WebArena confirm that AgentTrek's data generalizes well to unseen domains.

Table 5: Comparison of task success rate on WebArena

Model	WebArena
CodeLlama-7B-Instruct (Ou et al., 2024)	0.00
LLaMa3-chat-8B (Ou et al., 2024)	3.32
Qwen2.5-7B-Instruct	3.80
LLama3-chat-70B (Ou et al., 2024)	7.02
GPT-4o(Zhou et al., 2023)	13.10
GPT-4(Ou et al., 2024)	14.41
Synatra-CodeLlama-7B (Ou et al., 2024)	6.28
AutoWebGLM (OOD SFT) (Lai et al., 2024)	8.50
AutoWebGLM (In-domain RFT) (Lai et al., 2024)	18.20
Qwen2.5-7B-Instruct w/ **AgentTrek**	10.46
Qwen2.5-32B-Instruct w/ **AgentTrek**	**22.40**

ScreenSpot. Fine-tuning with the AgentTrek dataset significantly improved Qwen2-VL's grounding ability for both text and icon-based tasks, more than doubling its baseline performance and surpassing several models on the ScreenSpot benchmark. This demonstrates the strong impact of AgentTrek in enhancing the model's grounding capabilities for web-based GUI tasks.

Table 6: Comparison of grounding performance on ScreenSpot Web Grounding

Model	Text	Icon/Widget	Average
GPT-4 (Cheng et al., 2024)	9.2	8.8	9.0
GPT-4o (Cheng et al., 2024)	12.2	7.8	10.1
Qwen2-VL-7B	35.2	25.7	30.7
SeeClick (Cheng et al., 2024)	55.7	32.5	44.7
CogAgent (Cheng et al., 2024)	70.4	28.6	50.7
GPT-4 + OmniParser (Lu et al., 2024)	81.3	51.0	67.0
Qwen2-VL-7B w/ **AgentTrek**	**81.7**	**51.5**	**67.4**

Mind2Web. The baseline Qwen2-VL-7B model is excluded due to its poor ability to locate target elements, a key requirement for web tasks. Only the fine-tuned models are included in the results. Fine-tuning with the AgentTrek dataset significantly improved Qwen2-VL's performance, especially in the Operation F1 metric, where it outperformed both GPT-3.5 and GPT-4 in all settings. Combining AgentTrek with Mind2Web data produced the best results across all metrics. While fine-tuning with Mind2Web alone shows strong performance, the addition of AgentTrek further enhances the model. This demonstrates how AgentTrek provides grounded data, while Mind2Web adds resources for handling complex web tasks. Additional details on result sources are in Appendix J.2.

4 ANALYSIS

With our `AgentTrek` pipeline, we generate large-scale trajectory data excelling in three areas. First, the dataset is diverse, covering various domains and task types, and improved by internet-sourced tutorials, boosting agent performance by 230% when following detailed instructions. Second, the data is collected from real-world web environments, starting with RedPajama, filtering

Table 7: Performance comparison across different methods and evaluation settings. 'H', 'I', 'AT', 'M2W' stand for HTML, Image, AgentTrek, Mind2Web

Obs	Model	Method	Cross-Task			Cross-Website			Cross-Domain		
			Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR
HTML	GPT-3.5	Choice	19.4	59.2	16.8	14.9	56.5	14.1	25.2	57.9	24.1
	GPT-4	Choice	40.8	63.1	32.3	30.2	61.0	27.0	35.4	61.9	29.7
H + I	GPT-4	Choice	46.4	73.4	40.2	38.0	67.8	32.4	42.4	69.3	36.8
	GPT-4	SoM	29.6	-	20.3	20.1	-	13.9	27.0	-	23.7
Image	Qwen2-VL										
	+ AT	Vision	45.5	84.9	40.9	40.8	82.8	35.1	48.6	84.1	42.1
	+ M2W	Vision	54.8	89.5	50.9	52.9	83.9	44.9	51.8	86.8	47.7
	+ AT + M2W	Vision	60.8	88.9	55.7	57.6	88.1	51.4	56.0	87.5	52.6

23,430 tutorials to produce 10,398 successful trajectories from 127 websites. Third, the data captures both high- and low-level task details, including DOM/HTML structures, AXTree snapshots, video recordings, and screenshots. This comprehensive data enhances agent performance on long-horizon tasks, with a cost of just $0.551 per trajectory, making our pipeline efficient and scalable.

4.1 Importance of Tutorials

Tutorials extracted from the internet are key to guiding the replay process. First, they add diversity to the generated trajectories, as tutorials often offer different methods for achieving the same goal. Second, tutorials improve the agent's execution. We tested the agent on 400 tasks, replaying them twice—once with tutorials and once with only high-level goals. The results showed that detailed instructions boosted performance significantly. Without tutorials, only 63 effective trajectories were produced (15.78%), while with tutorials, the agent generated 208 effective trajectories (52%), a 230% improvement. For further analysis, see Appendix B.

4.2 Data Composition

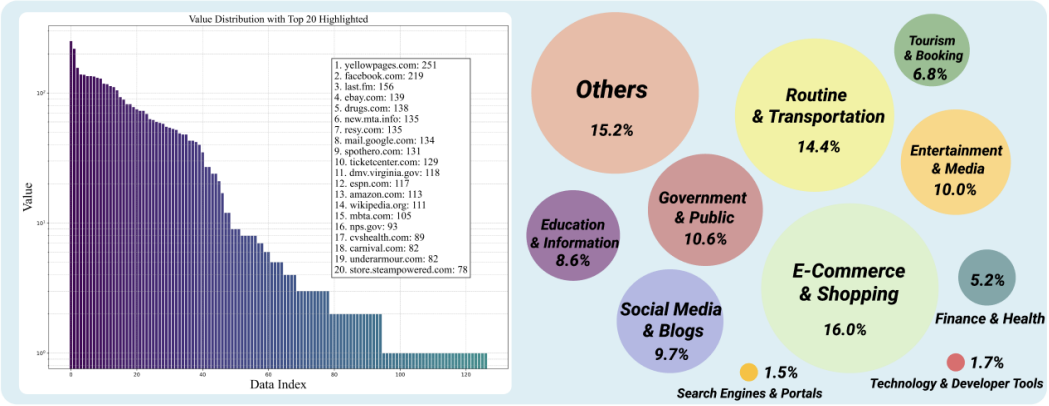

Figure 7: The distribution of website with domains involved in our dataset

To summarize the data flow through our pipeline: First, we filter tutorial data from the RedPajama web snapshot, then paraphrase it for clarity and classification. Next, we gather up-to-date data from mainstream websites for replay and collect effective trajectory data from the replays.

After filtering RedPajama's vast dataset, we retained over 18.8 million entries. By applying criteria such as recency and popularity, 23,430 tutorials were prepared for replay. With a success rate of 44.4%, we generated 10,398 trajectories, covering 127 websites across 11 distinct categories.

4.3 Comparison with Existing Work and Research Challenges

AgentTrek generates large-scale trajectory data with several key strengths, as shown in Table 1 (Niu et al., 2024; Lù et al., 2024; Deng et al., 2024; Yao et al., 2022; Song et al., 2024; Wornow et al., 2024). Our dataset includes nearly 5k verified trajectories, averaging 12.1 steps per trajectory, making it ideal for training and evaluating agents on long-horizon web tasks. It also stands out for its comprehensiveness, incorporating DOM/HTML structures, AXTree data, intermediate reasoning, full video recordings, and screenshots for each action.

Despite being fully automated, our dataset remains diverse, spanning 120 websites and 12 task categories. By using modern LLMs, we extract high-level objectives and detailed instructions, offering flexibility for future applications. Our pipeline significantly reduces the cost and scalability issues of human-annotated data, with a per-trajectory cost of just $0.551, making it efficient and scalable for large-scale data generation. Cost details are in C.

5 Related Work

LLM-based Agents. LLM-based agents are autonomous systems that leverage large language models (Brown et al., 2020) to interact with real-world websites and OS environments. These agents can understand natural language instructions and perform a wide range of complex tasks across various domains, such as e-commerce, online assistance, and knowledge navigation (Nakano et al., 2021; Cheng et al., 2024). Recent efforts in this space include models like SeeAct (Zheng et al., 2024) and WebVoyager (He et al., 2024), which aim to generalize agent behavior to real-world websites. While LLM-based agents have shown promise, challenges remain in the need for agent-specified data. Our work extends this line of research by introducing a cost-effective pipeline to generate comprehensive agent trajectory data, advancing the state-of-the-art in data synthesis for agent-based applications.

Agent Data. As the use of agents increases, the demand for scalable data also rises. Most existing datasets are designed for benchmarks (Zhou et al., 2023; Li et al., 2023; Deng et al., 2024) and often rely on human annotation, which limits scalability. Our pipeline addresses this issue by automatically generating agent trajectory data in a cost-effective manner. Recent works, such as NNetNav (Murty et al., 2025) and BAGEL (Murty et al., 2024), also tackle this challenge by creating synthetic trajectories without human supervision.

Automatic Evaluation for Digital Agents. Recently, there has been growing interest in automating the evaluation of digital agents using Vision-Language Models (VLMs) and Large Language Models (LLMs). These methods leverage models to assess agent performance in real-world tasks. Research in this area spans several dimensions: some works focus on trajectory-level success (Pan et al., 2024), while others evaluate stepwise success based on adherence to instructions (Wornow et al., 2024). Additionally, evaluations are conducted across various task environments, such as web-based platforms and mobile operating systems like Android and iOS (Pan et al., 2024). In our work, we prompt a VLM, GPT-4o, as an autonomous evaluator, using the agent's interaction process as inputs to assess whether the agent has successfully completed tasks at the trajectory level.

6 Conclusion

In this work, we introduce AgentTrek, an efficient pipeline designed to automatically generate comprehensive and cost-effective agent trajectory data. Additionally, we present a large and diverse dataset generated using this approach, which we validate by training models and evaluating their performance with promising result.Our research establishes a novel and promising direction for the future development of LLM agent, particularly in the automatic and low-cost synthesis of trajectory data. AgentTrek serves as a strong standard for agent data generation, setting the stage for future advancements in this field.

ACKNOWLEDGEMENT

The authors of this paper were supported by the ECS (27212023) from RGC of Hong Kong.

REFERENCES

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. *ArXiv preprint*, abs/2401.10935, 2024. URL https://arxiv.org/abs/2401.10935.

Together Computer. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim Alabdulmohsin, Avital Oliver, Piotr Padlewski, Alexey Gritsenko, Mario Lučić, and Neil Houlsby. Patch n' pack: Navit, a vision transformer for any aspect ratio and resolution, 2023. URL https://arxiv.org/abs/2307.06304.

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

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam Hadj Laradji, Manuel Del Verme, Tom Marty, L'eo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? *ArXiv preprint*, abs/2403.07718, 2024. URL https://arxiv.org/abs/2403.07718.

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

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, 2017. URL https://aclanthology.org/E17-2068.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. Autowebglm: A large language model-based web navigating agent, 2024. URL https://arxiv.org/abs/2404.03648.

Tao Li, Gang Li, Zhiwei Deng, Bryan Wang, and Yang Li. A zero-shot language agent for computer control with structured reflection. *ArXiv preprint*, abs/2310.08740, 2023. URL https://arxiv.org/abs/2310.08740.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on computer control agents, 2024. URL https://arxiv.org/abs/2406.03679.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent, 2024. URL https://arxiv.org/abs/2408.00203.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. Weblinx: Real-world website navigation with multi-turn dialogue, 2024.

Shikhar Murty, Christopher Manning, Peter Shaw, Mandar Joshi, and Kenton Lee. Bagel: Bootstrapping agents by guiding exploration with language, 2024. URL https://arxiv.org/abs/2403.08140.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild, 2025. URL https://arxiv.org/abs/2410.02907.

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

Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. Screenagent: A vision language model-driven computer control agent. 2024.

OpenAI. Gpt-4v(ision) system card, 2024. URL https://openai.com/research/gpt-4v-system-card.

Tianyue Ou, Frank F. Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta, Dan Roth, Graham Neubig, and Shuyan Zhou. Synatra: Turning indirect knowledge into direct demonstrations for digital agents at scale, 2024. URL https://arxiv.org/abs/2409.15637.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv preprint*, abs/2203.02155, 2022. URL https://arxiv.org/abs/2203.02155.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *ArXiv preprint*, abs/2404.06474, 2024. URL https://arxiv.org/abs/2404.06474.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *ArXiv preprint*, abs/2304.03277, 2023. URL https://arxiv.org/abs/2304.03277.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023. URL https://arxiv.org/abs/2307.16789.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control, 2023. URL https://arxiv.org/abs/2307.10088.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. 2024.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *ArXiv preprint*, abs/2409.12191, 2024. URL https://arxiv.org/abs/2409.12191.

Michael Wornow, Avanika Narayan, Ben Viggiano, Ishan S. Khare, Tathagat Verma, Tibor Thompson, Miguel Angel Fuentes Hernandez, Sudharsan Sundar, Chloe Trujillo, Krrish Chawla, Rongfei Lu, Justin Shen, Divya Nagaraj, Joshua Martinez, Vardhan Agrawal, Althea Hudson, Nigam H. Shah, and Christopher Re. Do multimodal foundation models understand enterprise workflows? a benchmark for business process management tasks. *ArXiv preprint*, abs/2406.13264, 2024. URL https://arxiv.org/abs/2406.13264.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *ArXiv preprint*, abs/2404.07972, 2024. URL https://arxiv.org/abs/2404.07972.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chun yue Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *ArXiv preprint*, abs/2310.11441, 2023. URL https://arxiv.org/abs/2310.11441.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35, 2022.

Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. ZeroGen: Efficient zero-shot learning via dataset generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.emnlp-main.801.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users, 2023.

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *ArXiv preprint*, abs/2307.13854, 2023. URL https://arxiv.org/abs/2307.13854.

Table of Contents in Appendix

A Calculation of Other Trajectory Datasets **15**

B Analysis of the Effectiveness of Tutorials **15**

C Cost Details **15**

D Evaluator Alignment **15**

E Action Mapping **16**

F Experimental Results on Textual Data **16**

G Details in Collecting Tutorials **16**

G.1 Prefilter Function 16

G.2 LLM Labeler Prompt 18

G.3 Tagging & Paraphrasing Prompt and Format 18

H Examples of Failed Guided Replay Trajectories **20**

I Scaling up AgentTrek **21**

J Evaluation Benchmarks **21**

J.1 GUI Grounding Evaluation 21

J.2 Offline GUI Agent Evaluation 21

K Detailed Description of guided replay **21**

A CALCULATION OF OTHER TRAJECTORY DATASETS

- **RUSS:** Cited based on the data provided in the table from WebLINX (Lù et al., 2024).
- **ScreenAgent:** Statistics obtained from the dataset available at `https://github.com/niuzaisheng/ScreenAgent/tree/main/data/ScreenAgent/train`.
- **WebLINX:** Calculated based on the train set information from Table 8 in (Lù et al., 2024) and data on HuggingFace (excluding the "say" actions), resulting in a total of 18,249 non-say actions with 969 demos.
- **Mind2Web:** Statistics derived from `https://huggingface.co/datasets/osunlp/Mind2Web`, specifically from the training subset.
- **Webshop (agent-eto):** Data statistics sourced from `https://huggingface.co/datasets/agent-eto/eto-sft-trajectory`.
- **WonderBread:** Calculations based on data presented in (Wornow et al., 2024).

B ANALYSIS OF THE EFFECTIVENESS OF TUTORIALS

Key factors contributing to this improvement include:

1. **Direct Access to Target URL:** Tutorials provide the target URL, allowing direct access to the initial task state, reducing errors in locating the correct webpage.
2. **Assisted Planning with Human Expertise:** Tutorials aid in planning by providing steps informed by human experience, which tend to be reliable, thereby reducing the likelihood of errors during task execution and bridging the gap in the agent's knowledge for unknown tasks.
3. **Navigating Multi-Level Menus:** Tutorials offer clear paths to hidden elements, preventing the agent from failing due to incorrect navigation through complex menus.

C COST DETAILS

In this part we provide the details of our cost in generating trajectory data with via our pipeline:

Phase	Cost per 1,000 Entries (USD)	Model Used
Tag and Paraphrase	0.886	gpt-4o-mini
Replay	215.359	gpt-4o-2024-08-06
Evaluator	3.104	gpt-4o-2024-08-06

Table 8: Cost breakdown for each phase in the process

Another two important factors are the ratio of web-related tutorials (0.275) and the Replay Success Rate (39.9%). Using these, we can calculate the cost per verified effective trajectory as follows:

$$\text{Cost per trajectory} = \frac{\text{Tag and Paraphrase price}}{\text{Web ratio}} + \frac{\text{Replay price} + \text{Evaluate price}}{\text{Replay Success Rate}}$$

The cost per 1,000 verified effective trajectories is 550.75 $.

D EVALUATOR ALIGNMENT

In this part, we provide the details of metrics between the human and automatic evaluator.

Trajectory	Evaluator	Accuracy
Web Tutorials	VLM Evaluator	84.0%
Webarena	GPT-4V	80.6%
	Captioner + GPT-4	82.1%
	Captioner + Mixtral	74.4%

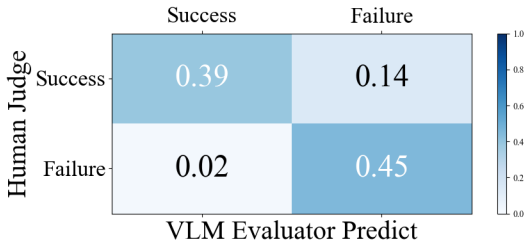

Figure 8: Confusion Matrix of our VLM evaluator's performance on the human-annotated validation set, compared with evaluators across different scenarios.

E ACTION MAPPING

Table 9: Mapping between Playwright and PyAutoGUI Action Spaces.

Category	Playwright Action	PyAutoGUI Action
Basic Actions	`page.click()`	`pyautogui.click()`
	`page.type()`	`pyautogui.write()`
	`page.press()`	`pyautogui.press()`
	`page.hover()`	`pyautogui.moveTo()`
	`page.scroll()`	`pyautogui.scroll()`
Advanced Actions	`page.fill()`	`pyautogui.write()` (clearing)
	`page.dblclick()`	`pyautogui.doubleClick()`
	`page.dragAndDrop()`	`pyautogui.dragTo()`

	`page.clear()`	`pyautogui.click()`
		`pyautogui.hotkey(ctrl, A)`
		`pyautogui.press(delete)`

Plugin	`playwright.select_option()`	`browser.select()`

F EXPERIMENTAL RESULTS ON TEXTUAL DATA

To provide further supports for the effective of our AgentTrek data, we conducted an experiment to evaluate the performance of a pure textual agent using the textual version data of our AgentTrek trajectories. This allow us to study the contribution of textual modalities of AgentTrek.

We fine-tuned the Qwen2.5-7B-Instruct model using AgentTrek trajectories that included accessibility tree as observations and playwright actions as the agent's action space. We then evaluated the model on WebArena, an OOD web agent benchmark featuring self-hosted websites. These websites are entirely out-of-domain (OOD) from the AgentTrek dataset, ensuring that the evaluation reflects the model's generalization capability.

We fine-tuned **Qwen2.5-7B-Instruct** on AgentTrek's textual data and achieved the following results on WebArena and Miniwob++ as shown in Table 5. We observe that our fine-tuned model achieves the highest performance among open-source web agents and approaches the performance of GPT-4o, demonstrating the effectiveness of AgentTrek data in improving real-world web agent capabilities and generalization across modalities.

G DETAILS IN COLLECTING TUTORIALS

G.1 PREFILTER FUNCTION

- **Keyword Density:** The web content must contain a minimum of 20 common keywords, ensuring sufficient topic coverage.
- **Keyword Diversity:** The text must incorporate at least 4 distinct common keywords.

System Prompt

```
You are an expert in evaluating the performance of a web navigation
agent.  The agent is designed to help a human user navigate a website
to complete a task.  Given the user's task goal, the agent's
trajectory, your goal is to decide whether the agent's execution is
successful or not.

*Evaluation Criteria*
Whether the agent's trajectory is effective and corresponding to the
goal

*Instructions*
1.  Review the agent's actions and reasoning processes step by step.
2.  if the agent is stuck in the very first login stage, which means
it fails to log into target website at the beginning, that's a
failure.
3.  Determine if the agent has achieved the task goal based on the
trajectory.  A task can be considered successful if most trajectory
is effective.
4.  the agent sometimes can't stop after finishing a task and
continue doing repeated actions.  these actions may be some failed
attempt after a series of correct actions.  the task should be
regarded as successful if the correct actions are effective and
almost reach the goal.
5.  if the agent is stuck in the loop at the early stage of the task,
which means they don't even get close to the goal before they get
stuck in the loop, that's a failure.  for example, the agent begin to
get stuck before third step.
6.  when the task is to change the google account password, it can't
be regarded as successful when agent finish at trying to click
"manage your account".
7.  if there are over 8 correct action in the trajectory, it can be
regard as a successful agent.
8.  final saving action is not a must.  the task is successful if the
agent does most things right and just forget to save the change at
last.
9.  if the original task has 2 subtasks, the agent only complete one
of them, that's still a success.  e.g.  the task is to update name
and birthday, but agent only update name, that's fine.
10.  if the task is to post a review, the agent can be considered
successful when it finish writing the review and reach the step to
post it, don't have to click the post button.
11.  Since we don't have a printer, some printing related task can be
considered successful if the agent reach the step to click print
button.
12.  if the task is finished at the initial state and the agent do
nothing because of it, it should also be regarded as successful.

*IMPORTANT*
1.  in the trajectory, an action always follows a corresponding
reasoning, which shows the observation and thought of the agent.
2.  your response should be contain:
Thoughts:  <your thoughts and reasoning process>
Status:  "success" or "failure"
```

User Prompt

```
The goal of the task:  {task_des}
trajectory:  {trajectory}
```

Figure 9: Prompts to query the VLM Autonomous Evaluator.

- **Essential Keyword Frequency:** At least one mandatory keywords must appear multiple times (minimum twice) within the content, demonstrating topic relevance.

G.2 LLM LABELER PROMPT

To achieve more precise and context-aware labeling, we designed the following prompt to guide the LLM in further assessing whether the given URL and context meet our requirements, as illustrated in Fig 10.

System Prompt

```
You are an assistant that classifies content based on specific
criteria.  Your task is to evaluate whether a given piece of content
serves as a tutorial specifically related to graphical user
interfaces (GUI), such as for web applications, desktop applications,
or operating systems.
```

Classification Criteria

```
The content qualifies as a GUI-related tutorial if it meets the
following conditions:
1.  It includes a task description outlining what needs to be
achieved.
2.  It provides clear step-by-step instructions for interacting with
a GUI, such as:
    - Step 1:  Open the application
    - Step 2:  Navigate to the settings menu

Given the URL and context, determine if the content is a GUI-related
tutorial or not.  Output '1' if it is a GUI-related tutorial and '0'
if it is not.  Provide only the number as the output.
```

User Prompt

```
- URL: {url}
- Context:  {context}
```

Figure 10: User Prompt for Classifying GUI Tutorials

G.3 TAGGING & PARAPHRASING PROMPT AND FORMAT

Here we present the prompt designed to utilize LLM to help do the tagging & paraphrasing of the identified GUI-tutorial related context.

User Prompt

```
The following is a tutorial from the website.  It may contain several
tutorials.  Please extract the first tutorial only and format the
first tutorial according to the specified schema:

Text:  {context}

Schema:
{
    "platform":
    "Platform category (choose from:  macOS, Windows (Default if not
specified in the tutorial), Linux, Android, iOS)",
    "target type":
    "Type of platform (choose from:  Web browser, PC app, Mobile app,
PC operating system, Mobile operating system, where the tutorial's
steps are performed).  Tutorials that involve interacting with the
browser software itself, such as 'opening Chrome settings,' should be
classified as a PC app type.",
    "target object":
    "Specific name of the web browser or (non web browser)
applications or operating system where the tutorial's steps are
performed (e.g., Chrome browser (Default for browser and web
tutorial), Microsoft Excel (app name), Windows system settings)",
    "target web URL":
    "The exact URL of the web page where the tutorial's actions take
place, applicable only if the target object is a web browser (e.g.,
None, https://mail.google.com, https://www.amazon.com,
https://github.com).  Be careful, the URL provided at the beginning
is always not the URL where the tutorial's actions are about.  For
example, a tutorial from
https://abidakon.com/how-to-make-google-slide-vertical/ about
changing Google Slides, its target web URL should be
https://docs.google.com/presentation.",
    "task description":
    "Task description text (Provide a concise summary in one
sentence, including essential details)",
    "prerequisites":
     "Prerequisite text describing necessary conditions before
starting the task",
    "instructions":
    [
     "Step_1:  Instruction text describing the action to be taken",
     // Following instructions
    ]
    "instructions steps":
    "Total number of instructions steps",
    "expected_result":
    "Text describing the expected result after following the
instructions"
}
```

Figure 11: User Prompt for Extracting and Formatting GUI Tutorials

H EXAMPLES OF FAILED GUIDED REPLAY TRAJECTORIES

Figure 12: Replay agent was unable to complete booking before the actual date due to the tutorial expiration.

Table 10: Multimodal Mind2Web Step SR across varying amounts of training data from AgnetTrek (AT).

Data Amount	Cross-Task	Cross-Website	Cross-Domain
20%	36.1	35.5	39.5
40%	41.0	35.8	42.5
60%	41.6	37.2	42.8
80%	42.6	38.0	44.3
100%	42.6	37.5	45.0

I SCALING UP AGENTTREK

In this section, we further scale up the data amount of AgnetTrek (more than 10K trajectories) to explore the effectiveness of AgentTrek in large-scale size. We trained the model using varying proportions of the dataset (20% to 100%) and assessed its performance on Multimodal-Mind2Web across three splits. The results are presented in Table 10. We observe that the performance improves steadily as more data is used, with the best results achieved when using the full dataset. This underscores the value of scaling up AgentTrek in improving model effectiveness.

J EVALUATION BENCHMARKS

In this section, we introduce more details of evaluation benchmarks used in our work.

J.1 GUI GROUNDING EVALUATION

ScreenSpot. ScreenSpot (Cheng et al., 2024) is a benchmark developed specifically for GUI visual grounding tasks, featuring 1.2K single-step instructions along with the coordinates of target elements. The dataset includes diverse grounding instructions tailored for mobile, desktop, and web platforms and categorizes elements into text and icons/widgets. Two distinct assessment scenarios are utilized: (1) *Original Instructions*, where models directly execute grounding actions as per the provided instructions; and (2) *Self-plan*, where models are expected to formulate plans in natural language based on the original instructions before carrying out the grounding actions.

J.2 OFFLINE GUI AGENT EVALUATION

Multimodal-Mind2Web. We evaluated the offline planning capabilities of GUI agents on websites using the Multimodal-Mind2Web benchmark (Zheng et al., 2024), which is an extension of the original Mind2Web benchmark (Deng et al., 2024). Performance was measured using Element Accuracy (Ele.Acc), Operation F1 (Op.F1), and Step Success Rate (Step SR).

The GPT-3.5 and GPT-4 results for the HTML and HTML+Image observation are derived from the SeeAct (Zheng et al., 2024) method. For the Choice method, it employs a DeBERTa-base cross-encoder to rank the interactive elements on the current HTML page. The top 50 elements are selected as options, and the GPT-3.5/GPT-4 model then chooses one of these elements as the answer. For the SoM method (Yang et al., 2023; Zheng et al., 2024), it renders a new webpage image by adding red bounding boxes and labels to every HTML node in the source code, allowing GPT-4 to understand the webpage screenshot and identify the target action object by referring to the labels. For the Image observation, as detailed in Section **??**, we use Qwen2VL (Wang et al., 2024) within a pure vision framework. Specifically, Qwen2VL processes only the webpage screenshots and specifies the target action object by generating its coordinates.

K DETAILED DESCRIPTION OF GUIDED REPLAY

In this section, we detailedly describe an example of model execution in guided replay.

Observation Prior to Execution Before executing any actions in the task execution process, the model observes the current webpage within the BrowserGym environment. Each webpage provides rich data, including its HTML structure, accessibility tree (AXTree), and screenshots. The model uses the AXTree as the primary observation source, with each element in the AXTree uniquely identified by a `[bid]`. This structured observation ensures accurate and consistent interaction:

Axtree with Element bid

```
[119] link 'Magento Admin Panel'
  [120] image 'Magento Admin Panel'
[121] navigation ''
  [122] menubar '', orientation='horizontal'
    [124] link '\ue604 DASHBOARD'
    [127] link '\ue60b SALES'
    ...
[614] banner ''
  [617] heading 'Dashboard'
  [620] link '\ue600 admin'
  ...
```

Figure 13: Observation Prior to Execution in Guided Replay

the information in tutorial the tutorial may provide a detailed textual description of the target element. The model realize that the target element is menubar, which associates with bid(122)

Axtree with Element bid

```
Step 1: click the menubar to see the sales
```

Figure 14: Observation Prior to Execution in Guided Replay

the action executed When the model performs an action, it does not need to provide fine-grained target element information such as coordinates, only the action type and the target object's bid as follows:

Axtree with Element bid

```
click('119')
```

Figure 15: Observation Prior to Execution in Guided Replay

We can retrieve the target elements in the webpage through the bid and perform corresponding operations through playwright action.