# AgentTrek: Agent Trajectory Synthesis via Guiding Replay with Web Tutorials

**Anonymous authors**
Paper under double-blind review

## 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 GUI 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.

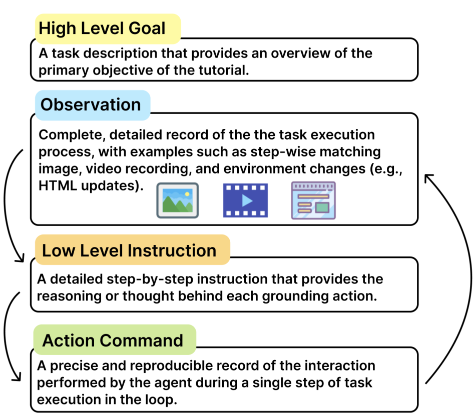


Figure 1: Expected GUI agent trajectories

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., 2024b), 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 et al., 2024). Their architecture and training paradigms are 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 GUI 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).

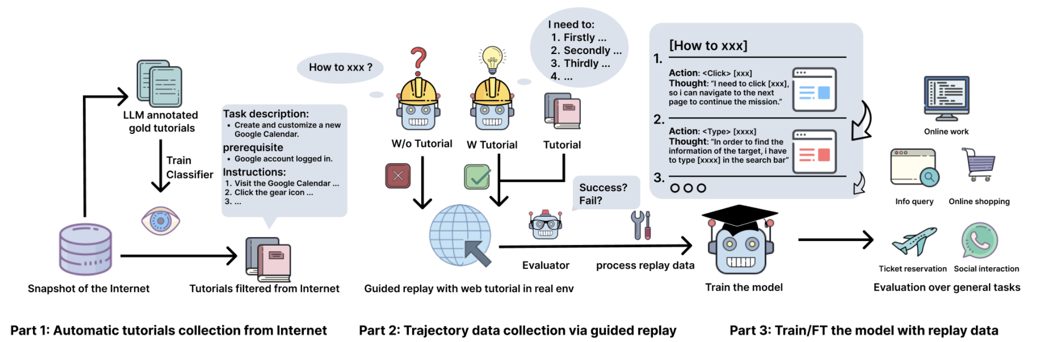


Figure 2: **Overview of the `AgentTrek` Pipeline**: (1) *Automatic Tutorial Collection from the Internet:* Tutorial-related data is extracted and filtered from internet sources using heuristic methods and a FastText model. An LLM processes the filtered textual data, transforming it into structured tutorials. (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.

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., 2023; 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 GUI 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 GUI agent trajectories.

Our experimental results demonstrate that training GUI agent models with these synthesized 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 GUI 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 obstacles 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.1.

| 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)** | **4902** | **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 visual language models (VLMs) 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 first extract web interaction tutorials from Redpajama dataset (Computer, 2023). A rule-based heuristic filter is applied to create a preliminary dataset, a subset of which is annotated by an advanced LLM to generate labeled samples for training a effective FastText classification model (Joulin et al., 2017), the tutorial classifier. This classifier further enhances the data quality through filtering. Finally, LLMs are employed to tag and paraphrase the raw text of tutorials into a standardized format, preparing them for the replay phase in Section 2.2.

#### 2.1.1 PREFILTER FUNCTION

Although GUI tutorials are abundant online, they constitute only a small fraction of web content, making a pre-filter essential for identifying relevant content. Similar patterns often appear in tutorials, such as distinctive keywords like 'click' and 'type', as well as platform-specific terms like 'macOS' and 'Windows'. We compiled a rule-based filter using keyword lists sourced from official websites and forums. Leveraging RedPajama data with over 20 billion URLs, our pre-filter applies **Keyword Matching** in the first 38k words, evaluates samples based on **Length**, and filters them 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

While initial rule-based filtering narrows the context, the proportion of true positive tutorial content remains low. To improve the quality and relevance of selected tutorials, we leverage an advanced LLM, GPT-4O MINI, for automated labeling, due to its strong ability to comprehend and analyze complex, information-dense text. Prior to full implementation, we tested GPT-4O MINI on a manually annotated ground-truth validation set, where it achieved an F1 score nearly 90%. In cases where human and LLM annotations conflicted, the LLM demonstrated the ability to identify tutorial-related

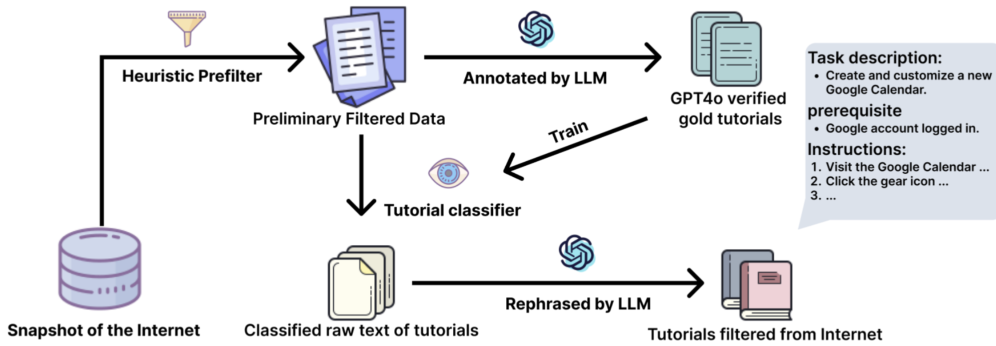


Figure 3: **Overview of the tutorial filtering and classification pipeline.** Starting with Redpajama, the data is prefiltered, annotated by an advanced LLM, and used to train a tutorial classifier. The classifier further filters the raw text, which is then paraphrased into structured tutorials with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

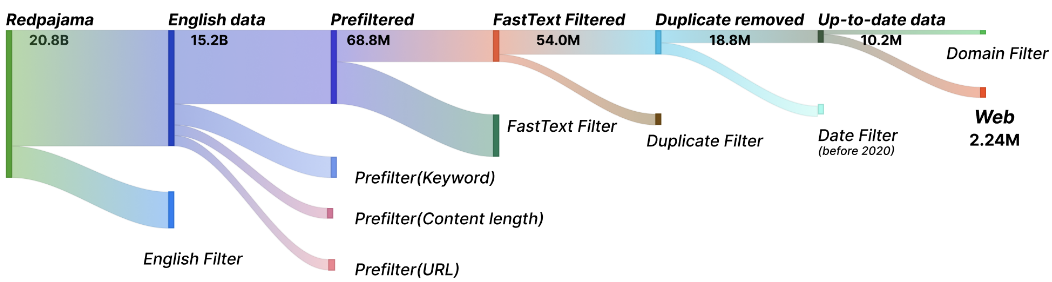


Figure 4: The data flow during the early stages of our pipeline.

content in lengthy texts that humans might overlook. This, along with the validation set result, suggests that GPT-4O MINI may surpass human performance in webpage labeling, enabling efficient generation of a large labeled dataset for training in the following section.

#### 2.1.3 FASTTEXT FILTER

Following the automated labeling process, we employed FastText, an n-gram-based deep learning model, as our classifier. FastText classifies tutorial text segments as tutorial or non-tutorial, with a binary output and a confidence score to enhance the accuracy of tutorial selection. To train the model, we combined LLM-labeled data with human-labeled samples, creating a dataset of approximately 90,000 examples. The train-test split is 95:5, with the model demonstrating strong classification performance. Using this classifier, we further curated the initial filtered dataset, collecting approximately 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 the raw tutorial content using the FastText model, we then tag and paraphrase the content for further processing, including extracting meta-information and formatting the tutorials according to a standardized template. To handle the length and noise of the raw tutorial data, we employed GPT-4O MINI, streamlining the tagging and paraphrasing while ensuring the output aligned with the comprehensive template and gold-standard examples.

The key components of the template include specifying the **Platform and Target** (e.g., macOS, Windows, browser, or app), providing a concise **Task Description**, listing **Prerequisites** needed before starting, outlining **Step-by-Step Instructions** for completing the task, and detailing the **Expected Outcome**. The cost for tagging and paraphrasing 1,000 entries is approximately 0.89 dollars.

### 2.2 TRAJECTORY DATA COLLECTION VIA GUIDED REPLAY

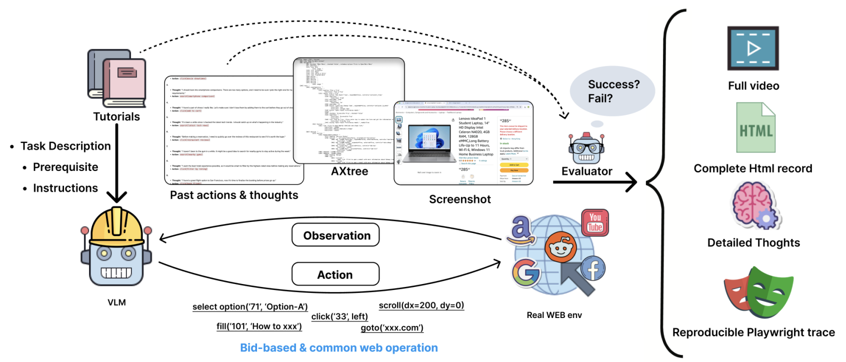


Figure 5: **Overview of Guided Replay data collection and evaluation pipeline.** A VLM agent is provided with the filtered and formatted tutorials, then observes and interacts with the real environment during execution, while all the actions and intermediate thoughts are recorded as data trajectory. 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 is designed to enhance an agent's web navigation capabilities by integrating high-level planning, low-level instructions, and grounded operations. Each data instance includes the following components:

**Task Information.** Detailed task metadata, including platform, task description, prerequisites, instructions, and expected outcomes, which support both planning and execution.

**Post-processed Textual Trajectory.** Refined after replay, highlighting key elements for model fine-tuning. This includes *Task Metadata*, summarizing the task to encourage adaptive decision-making, *Observations* to provide visual context, *Intermediate Reasoning* offering insights into the agent's decision-making process, and *Action Sequence* to capture detailed element information for web interactions.

**Screenshots and Video Recordings.** Visual records of the entire process for comprehensive documentation.

**Reproducible Native Trace.** Captured via Playwright, including DOM snapshots, HTML, network flow, and action sequences, allowing full reconstruction and detailed analysis of agent-environment interactions.

#### 2.2.2 GUIDED REPLAY WITH TUTORIALS

Although we have collected and processed high-quality tutorials, a significant gap remains in acquiring the grounding data crucial for training a more effective agent model. To address this, we leverage BrowserGym (Drouin et al., 2024a) to enable the model to replay tasks under the guidance of the generated tutorials.

BrowserGym is a versatile environment for web task automation in the Chromium browser, enabling Visual Language Model (VLM) agents to execute web-based operations (Drouin et al., 2024b). Agents are provided with tagged and paraphrased tutorials and a *target_web_url*, allowing them to

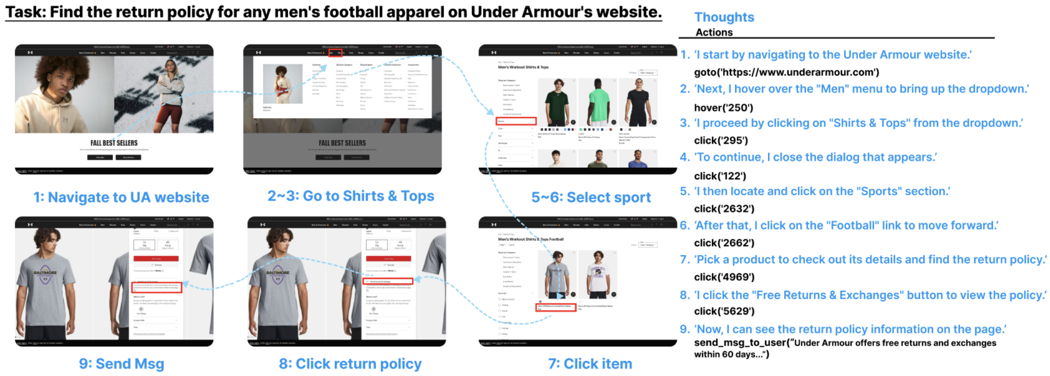


Figure 6: **Guided replay example.** This example demonstrates an agent's execution of finding the return policy for men's football apparel, showcasing its actions alongside the corresponding inner thoughts.

navigate directly to the task scene. Step-by-step instructions guide the agent through the task, with the expected outcome determining task completion.

The agent's initial observations include the webpage's viewport screenshot and accessibility tree (AXTree), but the HTML file is excluded due to its size and irrelevance to visual agents. Actions are executed using Playwright (Microsoft, 2023) functions such as $click$, $select\_option$, and $clear$, while Playwright also records detailed traces, including target elements, coordinates, screenshots, and DOM snapshots at the same time, along with agent's internal thoughts between actions.

Token consumption is about 8,027 per step and 86,114 per task. With GPT-4o-08-06, replaying 1,000 tasks costs approximately 215 dollars. Cost detail see A.3

### 2.2.3 Evaluation of Trajectory

Although a large amount of guided replay data has been recorded, it is crucial to extract the effective segments that can truly contribute to enhancing the agent's performance. Recent work by (Pan et al., 2024) highlights the potential of Visual Language Models (VLMs) in evaluating trajectory data using recorded images and interaction processes as input. VLMs are highly scalable, capable of processing large datasets concurrently at a low cost, and provide transparent evaluations. Therefore, we implemented a VLM Evaluator to further improve our data quality.

**VLM Evaluator Design.** To ensure trajectory data quality, we define *effectiveness* based on two criteria: adherence to task instructions and successful completion of core components. We employ GPT-4o as the backbone of our VLM evaluator, using a structured prompt to assess recorded trajectories. The evaluator receives the task description $d$, the agent's action history $\overline{a}$, and inner thoughts $\overline{r}$ for each step. The sequential format is: {task description; inner thought 1; action 1; inner thought 2; action 2; ...}, as illustrated in Figure 5. The VLM provides a trajectory-level assessment and performs stepwise analysis, offering justifications for any ineffective trajectory and identifying the earliest point of failure.

Table 3: Accuracy Comparison

| Trajectory | Evaluator | Acc. |
|---|---|---|
| Web Tutorials | VLM Eval | 84.0% |
| | GPT-4V | 80.6% |
| WebArena | Cap. + GPT-4 | 82.1% |
| | Cap. + Mixtral | 74.4% |

Table 4: Cost Breakdown

| Phase | Cost/1k ($) | Model |
|---|---|---|
| T&P | 0.89 | mini |
| Replay | 215.36 | 08-06 |
| Eval | 3.10 | 08-06 |
| **Total** | **219.35** | – |

**Validation on Human-Annotated Set.** Although the capabilities of Vision Language Models (VLMs) are well-recognized, validation is essential. To assess the automatic evaluator's performance, we manually reviewed 1,081 trajectories and created a validation set of 558 samples with human-annotated justifications.

As shown in the Table 3, despite handling different input formats across various task scenarios, the evaluator achieved high performance metrics. And according to the observationA.4, evaluator often applys stricter standards than human evaluators. This demonstrates its robustness in accurately identifying effective trajectories.

### 2.3 Train and Fine-tune the model with trajectory data

We chose a purely visual agent model that relies exclusively on screenshot-based observations, rather than incorporating accessibility trees or textual representations, for several reasons. First, GUIs are inherently visual, and mapping instructions to visual elements aligns more closely with human cognitive processes. Second, Textual representations, such as HTML or accessibility trees, are often verbose, which leads to heavy overhead for computation. Second, Different websites can have varying structures for their textual representation while image-based representations allow the model to unify its observations across diverse platforms with varying resolutions, improving generalization.

#### 2.3.1 Pure Vision & GUI Action Framework

In this work, we propose to unify observation and action space via pure vision and standard pyautogui commands with a pluggable action system.

Using pure vision as input eliminates the need for the model to understand different UI source codes across platforms, even when visually similar elements are written differently in HTML. Additionally, HTML input typically costs an average of 4,000 tokens per step. In contrast, recent VLMs with high-resolution multimodal understanding, such as Qwen2-VL, require only 1,200 tokens for a 720p image. This significantly lowers the computational cost while maintaining sufficient visual information for the task.

For action, we hoose the widely used standard pyautogui action space with a pluggable action system. Most web agent leverage playwright action sapce. But playwright actions incline to interact with html selector element instead of visual ui element. Therefore, we use pyautogui commands to unify basic GUI operations on web. Since we collect the data from website by playwright, we need to map the playwright actions to pyautogui actions as shown in the Figure 8. In addition, we utilize a pluggable action system to cover specific playwright action like select_option.

#### 2.3.2 Model Architecture and Training

Unlike agents that rely on structured UI representations like accessibility trees, vision-based grounding requires models to map intents directly to visual observations. For this, we chose Qwen2-VL (Wang et al., 2024b), which uses NaViT as an image encoder with dynamic resolution support (Dehghani et al., 2023). By removing absolute position embeddings and incorporating 2D-RoPE (Su et al., 2024), Qwen2-VL can process images of any resolution, efficiently converting them into variable visual tokens. This makes Qwen2-VL ideal for GUI agents, as it can encode high-resolution images with fewer token costs, making it well-suited for our tasks.

Our training process, starting with a VLM capable of high-resolution image understanding, consists of one tunnig stage. We use data from AgentTrek Data to enhance VLM capabilities in grounding and planning.

## 3 Experiments

AgentTrek automatically collects thousands of trajectories with detailed information, including inner thoughts and precise action coordinates, offering an ideal foundation for fine-tuning VLM into a reliable visual GUI agent. A successful visual GUI agent requires two key abilities: planning and grounding. We conducted extensive experiments to validate that AgentTrek data enhances the agent's capabilities in both areas.

### 3.1 Experimental Setup

To validate the effectiveness of our dataset, it is essential to demonstrate its impact on improving both the grounding and planning capabilities of the model. Therefore, we will evaluate the model's performance on benchmarks that assess these abilities.

For grounding ability, we employ the ScreenSpot benchmark (Cheng et al., 2024), which evaluates the model's single-step GUI grounding performance across various platforms. To assess planning ability, we utilize the Multimodal-Mind2Web benchmark (Deng et al., 2023), the multimodal extension of the web agent benchmark Mind2Web (Deng et al., 2024).

### 3.2 Web Grounding

ScreenSpot is a GUI visual grounding benchmark comprising 1.2K single-step instructions and target element bounding boxes. It spans mobile, desktop, and web environments, categorizing elements into text and icons. Given that our data originates exclusively from the web, we focus solely on web-based performance.

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 5: Comparison of grounding performance on ScreenSpot Web Grounding

| **Model** | **Text** | **Icon/Widget** | **Average** |
|---|---|---|---|
| GPT-4 | 9.2 | 8.8 | 9.0 |
| GPT-4o | 12.2 | 7.8 | 10.1 |
| Qwen2 VL | 35.2 | 25.7 | 30.7 |
| SeeClick | 55.7 | 32.5 | 44.7 |
| CogAgent | 70.4 | 28.6 | 50.7 |
| GPT-4 + OmniParser | 81.3 | 51.0 | 67.0 |
| Qwen2-VL w/ **AgentTrek** | **81.7** | **51.5** | **67.4** |

### 3.3 Web Planning

The baseline Qwen2-VL model is excluded due to its poor performance in locating target elements, a key requirement for web-based tasks. Thus, only the fine-tuned versions are presented in the table.

Fine-tuning with the AgentTrek dataset significantly improved Qwen2-VL's performance, particularly in the Operation F1 metric, where it outperformed both GPT-3.5 and GPT-4 across all settings.

The combination of AgentTrek and Mind2Web datasets yields the best performance across all metrics and settings. Although the model achieve great performance after fine-tunning with mind2web dataset, it can still benefits from the AgentTrek Data.

These results highlight the complementary nature of the two datasets: AgentTrek provides high-quality grounded data with precise coordinates, while Mind2Web offers valuable data for handling complex web-based tasks.

## 4 Analysis

With our `AgentTrek` pipeline, we generate large-scale trajectory data that excels in three areas. First, the dataset offers extensive diversity, covering multiple domains and task types, and benefiting from internet-sourced tutorials that enhance task execution. Our experiment showed a 230% performance increase when agents followed detailed instructions. Second, the data is gathered from real-world web environments, avoiding the limitations of simulations. Starting with RedPajama, we filtered and processed 12,526 tutorials, producing 4,902 successful trajectories from 127 websites.

Table 6: 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 |

Third, the data is comprehensive, capturing high- and low-level task details, including DOM/HTML structures, AXTree snapshots, video recordings, and screenshots. This rich data improves the agent's performance on long-horizon tasks, and with a per-trajectory cost of just $0.551, our pipeline offers an efficient, scalable solution for data generation.

## 4.1 Importance of Tutorials

Tutorials extracted from the internet play a crucial role in guiding the replay process. First, they ensure diversity in the generated trajectories. Tutorials often have distinct task goals, and even when they target the same objective, they may offer different execution methods, enriching the trajectory data. Second, tutorials significantly improve the agent's execution. We tested the agent on 400 tasks, replaying them twice: once with tutorials and once using only high-level goals. The results show that step-by-step instructions greatly enhanced performance. Without tutorials, only 63 effective trajectories were generated (15.78% of the total). With tutorials, the agent produced 208 effective trajectories (52%), marking an increase of over 230%, demonstrating the importance of detailed instructions in improving reliability and effectiveness. Analysis seeA.2

## 4.2 Data Composition

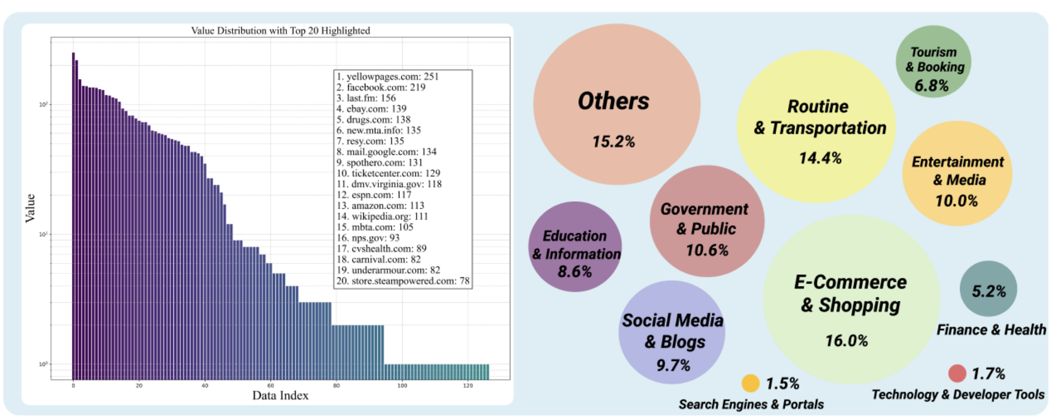


Figure 7: The distribution of website with domains involved in our dataset

To summarize the data flow through our pipeline: First, we filter tutorial data from the RedPajama web snapshot. Next, the filtered data is paraphrased for clarity and classification. We then gather up-to-date data from mainstream websites for replay and, finally, 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, 12,526 tutorials were prepared for replay. With a success rate of 39.1%, we generated 4,902 trajectories, covering 127 websites across 11 distinct categories.

### 4.3 COMPARISON WITH EXISTING WORK AND RESEARCH CHALLENGES

AgentTrek generates comprehensive, large-scale trajectory data, excelling in several key areas 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., 2024a). First, with nearly 5k verified trajectories and an average of 12.1 steps per trajectory, our dataset provides a strong foundation for training and evaluating agents on long-horizon web tasks. Second, it is the most complete dataset to date, incorporating DOM/HTML structures, AXTree data, intermediate reasoning steps, full video recordings, and corresponding screenshots for each action.

Moreover, despite full automation without human intervention, our dataset maintains diversity across 120 websites and 12 distinct task categories. By leveraging modern large language models (LLMs), we can extract both high-level task objectives and detailed step-by-step instructions, offering flexibility for future use.

Finally, our pipeline significantly reduces the cost and scalability challenges of human-annotated data collection. With a success rate factored in, the cost per trajectory is just $0.551, making our approach both efficient and scalable for large-scale data generation. Cost detail see A.3

## 5 RELATED WORK

**LLM-based Agents.** LLM-based agents are autonomous systems that leverage large language models (Brown et al.) 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., 2024a) 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 agents gain increasing popularity, the demand for efficient and scalable data is becoming both larger and more urgent. However, most existing data primarily serve as supplements to various benchmarks (Zhou et al., 2023; Li et al., 2023; Deng et al., 2024), with few datasets specifically designed for agent trajectory analysis. Furthermore, these datasets are often limited by the need for human annotation, which hampers scalability. In our work, our pipeline managed to automatically generate comprehensive agent trajectory data in a cost-efficient manner, paving the way for a new direction in data synthesis within the field of agents.

**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., 2024b). 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 agent's interacton 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.

# REFERENCES


Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners.

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 arXiv:2401.10935*, 2024.

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

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023. URL `https://arxiv.org/abs/2306.06070`.

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 H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 21–27 Jul 2024a. URL `https://proceedings.mlr.press/v235/drouin24a.html`.

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*, abs/2403.07718, 2024b. URL `https://api.semanticscholar.org/CorpusID:268363855`.

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 arXiv:2401.13919*, 2024.

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*, pp. 427–431. Association for Computational Linguistics, April 2017.

Tao Li, Gang Li, Zhiwei Deng, Bryan Wang, and Yang Li. A zero-shot language agent for computer control with structured reflection. *arXiv preprint arXiv:2310.08740*, 2023.

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`.

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

Microsoft. Playwright for python documentation. `https://playwright.dev/python/`, 2023.

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 arXiv:2112.09332*, 2021.

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, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

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*, abs/2203.02155, 2022. URL `https://api.semanticscholar.org/CorpusID:246426909`.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *ArXiv*, abs/2404.06474, 2024. URL `https://api.semanticscholar.org/CorpusID:269009430`.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *ArXiv*, abs/2304.03277, 2023. URL `https://api.semanticscholar.org/CorpusID:257985497`.

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.

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 arXiv:2406.13264*, 2024a. URL `https://hazyresearch.stanford.edu/wonderbread-website`.

Michael Wornow, Avanika Narayan, Ben T 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*, abs/2406.13264, 2024b. URL `https://api.semanticscholar.org/CorpusID:270620942`.

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 arXiv:2404.07972*, 2024.

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*, 35:20744–20757, 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. *ArXiv*, abs/2202.07922, 2022. URL `https://api.semanticscholar.org/CorpusID:246867045`.

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 arXiv:2401.01614*, 2024a.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024b. 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 arXiv:2307.13854*, 2023.

# A Appendix

## A.1 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., 2024a).

## A.2 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.

## A.3 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 7: 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 $.

## A.4 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% |

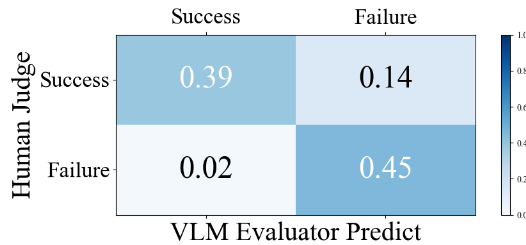


Figure 8: Confusion Matrix of our VLM evaluator's performance on the human-annotated validation set, compared with evaluators across different scenarios.

### A.5 Action Mapping

Table 8: Mapping between Playwright and PyAutoGUI Action Spaces.

| Category | Playwright Action | PyAutoGUI Action |
|---|---|---|
| Basic Actions | `page.click()`<br>`page.type()`<br>`page.press()`<br>`page.hover()`<br>`page.scroll()` | `pyautogui.click()`<br>`pyautogui.write()`<br>`pyautogui.press()`<br>`pyautogui.moveTo()`<br>`pyautogui.scroll()` |
| Advanced Actions | `page.fill()`<br>`page.dblclick()`<br>`page.dragAndDrop()` | `pyautogui.write()` (clearing)<br>`pyautogui.doubleClick()`<br>`pyautogui.dragTo()` |
| | `page.clear()` | `pyautogui.click()`<br>`pyautogui.hotkey(ctrl, A)`<br>`pyautogui.press(delete)` |
| Plugin | `playwright.select_option()` | `browser.select()` |

**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.