Lightweight Adaptive Topological Layout And Semantic Mapping in Vision-and-Language Navigation on Websites

Pingrui Lai[1], **Zihao Xie**[1], **Hua Yang**[1]*

[1]School of Information Science and Electronical Engineering & School of Integrated Circuits, Shanghai Jiao Tong University, Shanghai, China
laipingrui@sjtu.edu.cn, xzh031202@sjtu.edu.cn, hyang@sjtu.edu.cn

Abstract

Vision-and-Language navigation on websites requires agents to navigate target webpages and answer questions based on human instructions. Current web agents primarily leverage Large Language Models (LLMs) for semantic understanding and reasoning, but still suffer from limited navigation performance and slow inference speed. Constructing a global map across webpages can effectively enhance both navigation accuracy and efficiency. However, this is challenged by the open structure of web navigation graphs and the dynamic nature of web layouts. In this paper, we propose ATLAS: Adaptive Topological Layout And Semantic mapping, a framework that adaptively constructs a time-varying, unbounded topological map across webpages and unifies heterogeneous elements through semantic representation. This enables both global path planning and local element selection for web-based navigation and question answering. As a lightweight approach, ATLAS significantly outperforms existing state-of-the-art methods on the WebVLN benchmark with a 10% improvement in success rate, and achieves the highest average task success rate on both the Mind2Web and WebArena benchmarks.

Code — https://github.com/26aaai/ATLAS

Introduction

Every day, billions of users interact with the web—clicking links, filling out forms, and retrieving information. Automating such interactions is essential for building intelligent agents, yet enabling machines to navigate the web effectively remains a challenging task.

Current web navigation can be regarded as a vision-and-language navigation problem (Anderson et al. 2018), where agents follow instructions, comprehend the structural layout of webpages, and locate specific information accordingly. Previous approaches typically trained web agents using reinforcement learning (RL) or multimodal models. In recent years, with the rapid advancement of large language models (LLMs), recent works have attempted to leverage LLMs to build web agents capable of navigating webpages that follow natural language instructions. These agents have achieved promising results in web navigation tasks.

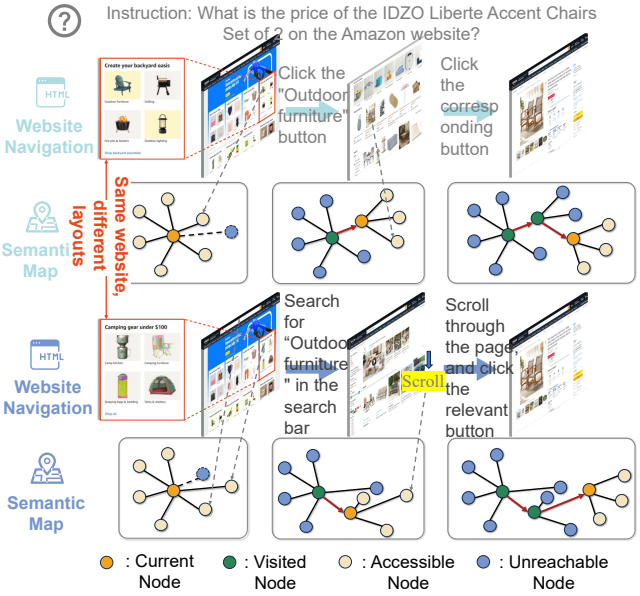

Figure 1: **An Example of the Dynamic Nature of Webpage Layouts.** Even the same webpage can vary structurally when accessed by different users or at different times. This results in a mismatch between theoretically reachable nodes and those that are practically accessible in the web semantic graph. The agent is supposed to make decisions based on varying observation spaces.

However, most existing web agents heavily rely on the memory and reasoning capabilities of LLMs. While the strength of pre-trained LLMs lies in their zero-shot generalization across tasks, two major limitations persist: (1) limited navigation performance and (2) slow inference speed. LLMs lack explicit modules for efficient state representation and action selection, and their token-by-token generation introduces significant latency. In terms of both task success rate and completion time, LLM-based web agents still lag far behind human performance, indicating that current LLM-centric approaches are insufficient for supporting efficient web navigation systems.

We observe that web navigation, like robot navigation, can be formulated as a Partially Observable Markov Deci-

*Corresponding author.

sion Process (POMDP) (Kaelbling, Littman, and Cassandra 1998). The action space of a web agent (e.g., clicking buttons or scrolling pages) is analogous to that of robot navigation (e.g., moving forward or turning left) where each action leads to a new observation space. From this perspective, one of the key factors that has significantly improved performance and efficiency in robot navigation is the construction of a global map during exploration (e.g., via Simultaneous Localization and Mapping, SLAM (Durrant-Whyte and Bailey 2006)), which provides a global view and structured environmental representation for decision-making.

However, applying global map construction to web navigation poses a significant challenge due to **the openness of the navigation graph structure**. Unlike navigation maps in the real world that typically exhibit well-defined boundaries, webpages contain a wide variety of elements such as buttons, hyperlinks, and content blocks, which lead to different target pages. This results in a connection graph that can expand indefinitely, making it nearly impossible for the agent to exhaustively explore all possible regions. Another key challenge lies in **the dynamic nature of webpage layouts**. As illustrated in Figure 1, even the same webpage may exhibit notable layout variations across different users or access times. Such variability introduces discrepancies between theoretically reachable nodes and those that are actually accessible in the current state, leading to a mismatch between the observation space and the action space. This further increases the complexity of the agent's perception and decision-making processes.

To address these challenges, we propose **ATLAS**: **A**daptive **T**opological **L**ayout **A**nd **S**emantic mapping. ATLAS facilitates the exploration process in navigation by progressively constructing a time-varying, unbounded topological map of the web environment, supporting both global path planning and local element selection. In addition, it introduces a semantic mapping module that unifies diverse HTML elements into a structured representation, enabling consistent scene understanding. Compared to existing methods, ATLAS is a lightweight approach that achieves superior navigation performance while significantly reducing inference latency. Specifically, ATLAS achieves state-of-the-art (sota) performance on WebVLN, Mind2Web, and WebArena datasets. In terms of inference speed, ATLAS runs 2 to 5 times faster than open-source LLMs (e.g., Llama 3). Our contributions are summarized as follows:

- We propose ATLAS, a lightweight web navigation framework that constructs adaptive topological layouts to enable more effective global and local planning.
- We design a semantic mapping module that transforms heterogeneous HTML elements into a unified representation, allowing the agent to reason over structured web content.
- ATLAS outperforms sota methods on the WebVLN benchmark by over 10% in success rate, and achieves 2–5× faster inference compared to open-source LLMs.

Related Work

Traditional Web Navigation Methods

Web navigation tasks (Shi et al. 2017; Yang et al. 2023) have evolved from simple scenarios—such as answering questions based on a single webpage (Chang et al. 2022)—to more complex, multi-step decision-making settings (Zhou et al. 2023). Early research leveraged RL to enable agents to explore web environments (Liu et al. 2018). Subsequent work (Jia, Kiros, and Ba 2019; He et al. 2021) extended capabilities by utilizing the Document Object Model (DOM) structure to perform more complex tasks such as booking flights. Other studies have expanded the scope from pure navigation to interactive operation tasks, involving both static page manipulations (Deng et al. 2023) and dynamic web interactions (Yao et al. 2022).

Recent approaches aim to unify web element understanding through language and multimodal models. For instance, WebAgent (Gur et al. 2023) pretrains a T5 model to extract HTML information and employs Flan-U-PaLM (Chowdhery et al. 2023) for agent construction. PIX2ACT (Shaw et al. 2023) takes webpage screenshots as input to predict agent actions end-to-end. WebGUM (Furuta et al. 2023) combines T5 (Raffel et al. 2020) with a Vision Transformer, jointly leveraging screenshots and HTML text for navigation. Although some systems have been successfully deployed on live websites (Zhou et al. 2023), there remains a significant performance gap between agents and human users (Lai et al. 2024), and effective strategies for further improving navigation performance are still lacking.

Web Navigation with Large Language Models

The strong reasoning capabilities of LLMs enable zero-shot performance on web navigation tasks. Accordingly, there has been growing interest in integrating LLMs with web environments. Early work such as WebGPT (Nakano et al. 2021) and WebGLM (Liu et al. 2023a) connected LLMs with web interfaces to perform question-answering (QA) tasks. Several efforts have focused on enabling LLMs to better handle structured web data. StructGPT (Jiang et al. 2023) explored ways to enhance zero-shot reasoning over structured information. MindAct (Deng et al. 2023) proposed a method to identify target elements via iterative filtering and multiple-choice selection mechanisms. (Kim, Baldi, and McAleer 2023) introduced a recursive prompting method to improve GPT-4's navigation capabilities. (Liu et al. 2023c) proposed a multi-agent collaboration framework to improve performance in the WebShop environment (Yao et al. 2022). (Zeng et al. 2023) fine-tuned LLaMA-2 on interaction trajectories and instruction data, achieving better performance than traditional agent-based baselines.

Recent work has shifted focus toward evaluating LLM-based agents on online websites. WebVoyager (He et al. 2024) constructed a multimodal agent capable of accomplishing tasks in real-world online environments. AutoWebGLM (Lai et al. 2024), based on ChatGLM, demonstrated strong web browsing capabilities. SeeAct (Zheng et al. 2024) conducted the first human evaluation on live websites. AgentTrek (Xu et al. 2024) utilized GPT-4o to filter

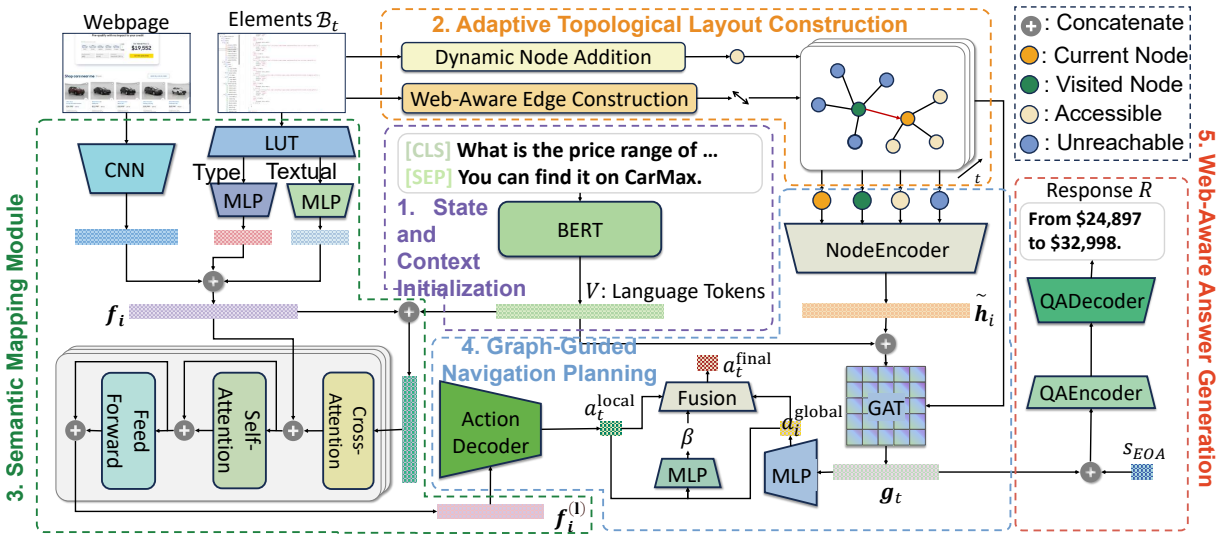

Figure 2: **Overview of ATLAS.** ATLAS comprises five main components: (1) State and Context Initialization, which encodes language tokens using a pre-trained BERT, (2) Adaptive Topological Layout Construction, which dynamically builds a web-oriented graph structure, (3) Semantic Mapping Module, which transforms heterogeneous HTML elements into unified semantic representations, (4) Graph-Guided Navigation Planning, which performs both global navigation and local element selection, and (5) Web-Aware Answer Generation, which leverages the constructed semantic graph for question answering.

low-quality trajectories using task descriptions, actions, and reasoning traces. Despite the promising potential of LLM-based web agents, several limitations remain. These include high response latency (e.g., WebVoyager suffers from token overload) and an overreliance on LLMs, which makes it difficult to achieve architectural improvements in navigation performance through either fine-tuning or prompt engineering alone (Xue et al. 2025).

Method

Problem Formulation

This task can be formulated as a POMDP, represented by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, \mathcal{R})$. Here, $\mathcal{S}$ denotes the set of environment states, with each state corresponding to the information of a specific webpage. At the beginning of each episode, the agent receives a natural language instruction Q and an auxiliary description D, which specify the navigation goal and relevant details. At each timestep t, the agent is in state $s_t \in \mathcal{S}$ and receives an observation $o_t \in \mathcal{O}$, where $o_t = (I_t, \mathcal{B}_t)$. I_t is an RGB screenshot of the current webpage, and $\mathcal{B}_t$ is the set of clickable buttons on the current page. Each button $b \in \mathcal{B}_t$ is described by a tuple (d_b, e_b), where d_b is a textual description (e.g., the HTML `alt` attribute), and e_b is an associated image if available. The agent selects an action $a_t \in \mathcal{A}$, where the action space $\mathcal{A}_t \subseteq \mathcal{B}_t \cup \{b_{[\text{EOA}]}\}$ consists of the subset of reachable clickable buttons and a special "stop" action $b_{[\text{EOA}]}$ indicating the end of navigation. After executing action a_t, the agent transitions to a new state s_{t+1} according to the transition function $T(s_{t+1} \mid s_t, a_t)$, which follows the corresponding structure of the website. The episode terminates when the agent selects the "stop" action or reaches a predefined maximum number of steps. Upon termination, the agent is required to generate a response or answer R to the initial question Q based on the information in the final state $s_{[EOA]}$. The objective is to learn a policy $\pi(a_t \mid h_t)$, where h_t is the history of observations and actions up to time t, so as to maximize the expected reward $\mathcal{R}$, which reflects both the success of navigation to the target webpage and the correctness of the generated answer.

Overview

As illustrated in Figure 2, ATLAS consists of five main components.

1. State and Context Initialization We utilize a pre-trained BERT model (Devlin et al. 2019) to initialize the model state and context tokens. At initialization ($t = 0$), the model receives a sequence consisting of the classification token [CLS], the separation token [SEP], and language tokens V extracted from the question Q and auxiliary description D. The embedded [CLS] token serves as the initial state representation s_0 and is continuously updated during navigation to maintain awareness of the overall task. The initialization process is defined as:

$$s_0, V = \text{Init}([\text{CLS}], Q, [\text{SEP}], D), \tag{1}$$

where $\text{Init}(\cdot)$ denotes the initialization function. The language tokens V are generated only once at initialization and are subsequently used as keys and values in the Transformer.

2. Adaptive Topological Layout Construction The adaptive topological layout module incrementally builds a web-oriented graph representation, capturing both the structural relationships between pages and the semantic coherence of

navigation paths. At each timestep t, we maintain a graph $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$ and construct the nodes and edges of the graph as follows:

Dynamic Node Addition The node set $\mathcal{V}_t$ contains three types of nodes: (1) visited nodes $\mathcal{V}_t^{\text{visited}}$, (2) accessible nodes $\mathcal{V}_t^{\text{accessible}}$, and (3) the current node v_t^{current}. Each node v_i is associated with a semantic representation $\mathbf{h}_i$ that encodes both visual and textual features. The computation of $\mathbf{h}_i$ depends on the node's state: $[a]$ For visited nodes. $[b]$ For accessible but unvisited nodes. $[c]$ For the current node. $\mathbf{h}_i$ is computed as:

$$\mathbf{h}_i = \begin{cases} \frac{1}{|\mathcal{O}_i|} \sum_{\mathbf{o} \in \mathcal{O}_i} \mathbf{o}, & \text{if } [a] \\ \frac{1}{|\mathcal{N}_i|} \sum_{v_j \in \mathcal{N}_i} \mathbf{o}_{j \to i}, & \text{if } [b] \\ \mathbf{o}_i^{\text{latest}}, & \text{if } [c] \end{cases} \tag{2}$$

where $\mathcal{O}_i$ denotes all past observations at node v_i, $\mathcal{N}_i$ is the set of neighboring nodes with partial views of v_i, and $\mathbf{o}_i^{\text{latest}}$ refers to the recent observation at the current node.

Web-Aware Edge Construction The edges between nodes in the graph are constructed by combining structural hyperlink information with semantic similarity. The edge weight e_{ij} between nodes v_i and v_j is defined as:

$$e_{ij} = \mathbf{1}_{\text{link}}(v_i, v_j) + \alpha \cdot \cos\big(\mathbf{h}_i, \mathbf{h}_j\big), \tag{3}$$

where $\mathbf{1}_{\text{link}}(v_i, v_j)$ is a binary indicator function defined as:

$$\mathbf{1}_{\text{link}}(v_i, v_j) = \begin{cases} 1, & \text{if there exists a link from } v_i \text{ to } v_j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

$\cos(\mathbf{h}_i, \mathbf{h}_j) = \frac{\mathbf{h}_i^\top \mathbf{h}_j}{\|\mathbf{h}_i\| \|\mathbf{h}_j\|}$ denotes the similarity between the semantic embeddings $\mathbf{h}_i$ and $\mathbf{h}_j$ of nodes i and j, respectively. α is a hyperparameter that balances the relative contributions of structural connectivity and semantic similarity.

Adaptive Graph Update When the agent visits a new webpage, the graph is updated as:

$$\mathcal{V}_{t+1} = \mathcal{V}_t \cup \{v_{t+1}^{\text{current}}\} \cup \mathcal{N}(v_{t+1}^{\text{current}}), \tag{5}$$

$$\mathcal{E}_{t+1} = \mathcal{E}_t \cup \{(v_t^{\text{current}}, v_{t+1}^{\text{current}})\} \cup \mathcal{E}_{\text{new}}, \tag{6}$$

where $\mathcal{N}(\cdot)$ denotes accessible neighboring webpages, and $\mathcal{E}_{\text{new}}$ denotes newly discovered edges.

3. Semantic Mapping Module Let $\mathcal{B}_t = \{b_1, \ldots, b_{|\mathcal{B}_t|}\}$ denote the set of HTML elements on page t. The semantic mapping module encodes each element $b_i \in \mathcal{B}_t$ into a unified semantic representation $\mathbf{f}_i \in \mathbb{R}^d$ through a sequence of modality-specific embeddings and transformer-based fusion.

Multi-Modal Embedding Each b_i is mapped into a type embedding vector based on its HTML tag and attributes:

$$\mathbf{t}_i = \phi_{\text{type}}(b_i) \tag{7}$$

where $\phi_{\text{type}}(\cdot)$ is a learnable function that combines a tag embedding lookup ϕ_{tag} and a MLP over selected attributes ϕ_{attr}, followed by a linear projection:

$$\phi_{\text{type}}(b_i) = W_t \cdot [\phi_{\text{tag}}(\text{tag}(b_i)), \phi_{\text{attr}}(\text{attributes}(b_i))] + \mathbf{b}_t \tag{8}$$

where W_t and $\mathbf{b}_t$ are learnable parameters of the linear projection. The final element representation $\mathbf{f}_i \in \mathbb{R}^d$ is obtained by concatenating its visual, type, and textual features:

$$\mathbf{f}_i = [\phi_{\text{vis}}(b_i), \phi_{\text{type}}(b_i), \phi_{\text{text}}(b_i)] \tag{9}$$

where $\phi_{\text{vis}} : \mathcal{B} \to \mathbb{R}^{d_v}$ extracts features from the bounding box region of element b, using a Convolutional Neural Network (CNN) encoder. $\phi_{\text{text}} : \mathcal{B} \to \mathbb{R}^{d_s}$ encodes the textual content using a Multilayer Perceptron (MLP) encoder.

Semantic Coherence To capture structural and contextual relationships among all elements on the webpage, we apply a multi-layer transformer encoder (SemanticEncoder) over the sequence of fused element embeddings:

$$[\mathbf{f}'_1, \ldots, \mathbf{f}'_{|\mathcal{B}_t|}] = \text{SemanticEncoder}([\mathbf{f}_1, \ldots, \mathbf{f}_{|\mathcal{B}_t|}]), \tag{10}$$

where each transformer layer alternates between:

$$\mathbf{z}_i^{(\ell)} = \text{MHA}\left(\mathbf{f}_i^{(\ell-1)}, \mathbf{F}^{(\ell-1)}, \mathbf{F}^{(\ell-1)}\right), \tag{11}$$

$$\mathbf{f}_i^{(\ell)} = \text{FFN}\left(\text{LayerNorm}(\mathbf{z}_i^{(\ell)})\right) + \mathbf{z}_i^{(\ell)}, \tag{12}$$

where MHA denotes multi-head attention, FFN is a feed-forward network, and $\mathbf{F}^{(\ell-1)}$ is the set of all element features from the previous layer. Cross-attention with instruction tokens V can be inserted between layers.

4. Graph-Guided Navigation Planning The agent performs hierarchical decision-making by leveraging both global topological reasoning and local element interactions.

Global Navigation Planning To reason over the topological structure of the web environment, we first encode all nodes $v_i \in \mathcal{V}_t$ in the graph $\mathcal{G}_t$ using a Transformer-based node encoder:

$$\widetilde{\mathbf{h}}_i = \text{NodeEncoder}(\mathbf{h}_i), \tag{13}$$

where $\mathbf{h}_i$ is the initial pooled representation of node v_i, as computed in the topological layout module. $\mathbf{h}_t^{\text{curr}}$ denotes the semantic embedding of the current node at time t, $\mathbf{g}_t$ is the global graph context at time t. These transformed node embeddings are then concatenated with V, and processed by a graph attention network (Veličković et al. 2017):

$$\mathbf{g}_t = \text{GAT}(\mathcal{G}_t, [\widetilde{\mathbf{h}}_i, V], \mathbf{h}_t^{\text{curr}}), \tag{14}$$

$$a_t^{\text{global}} = \text{FFN}([\widetilde{\mathbf{h}}_i, \mathbf{g}_t, \mathbf{h}_t^{\text{curr}}]), \tag{15}$$

where a_t^{global} means the global action, GAT$(\cdot)$ means Graph Attention Transformer, it applies graph attention conditioned on the node context and instruction tokens.

Local Element Selection For local decision-making, we apply a transformer-based action decoder over the previously computed semantic features $\mathbf{f}_i^{(\ell)}$:

$$a_t^{\text{local}} = \text{ActionDecoder}([s_t, \mathbf{f}_1^{(\ell)}, \ldots, \mathbf{f}_{|\mathcal{B}_t|}^{(\ell)}]), \tag{16}$$

where a_t^{local} means the local action.

Feature Fusion To integrate global planning with local evidence, we compute a fusion weight β_t and interpolate between global and local scores:

$$\beta_t = \sigma(\text{FFN}([\mathbf{g}_t, \mathbf{f}_1^{(\ell)}, \ldots, \mathbf{f}_{|\mathcal{B}_t|}^{(\ell)}]), \tag{17}$$

$$a_t^{\text{final}} = \beta_t \cdot a_t^{\text{global}} + (1 - \beta_t) \cdot a_t^{\text{local}}, \tag{18}$$

where σ is the sigmoid function, a_t^{final} is the fused action at timestep t after combining global and local evidence.

5. Web-Aware Answer Generation To generate answers grounded in both the navigation trajectory and the underlying web structure, we adopt a multi-stage process that fuses global semantic context with the language query.

Context Representation Construction We first construct a context representation that integrates the final navigation state s_{EOA} and the dynamically constructed topological graph $\mathcal{G}_{\text{final}}$:

$$\mathbf{c}_{\text{answer}} = \text{QAEncoder}([s_{\text{EOA}}, \mathbf{g}_{\text{final}}]) \tag{19}$$

where QAEncoder is a multi-layer transformer. It jointly encodes the fused sequence of the final navigation state, language context, and the web-aware topological embedding.

Answer Generation Given the encoded context $\mathbf{c}_{\text{answer}}$, we decode the final answer using a Transformer-based QADecoder:

$$R = \text{QADecoder}(\mathbf{c}_{\text{answer}}) \tag{20}$$

where QADecoder is a transformer decoder that generates the answer sequence conditioned on the context embedding.

Training Strategy

Our training objective consists of three major components: semantic understanding, navigation policy learning, and answer generation. The semantic mapping loss $\mathcal{L}_{\text{semantic}}$ captures the structural understanding of HTML content through three sub-losses:

$$\mathcal{L}_{\text{semantic}} = \mathcal{L}_{\text{type}} + \mathcal{L}_{\text{rel}} + \mathcal{L}_{\text{coh}}, \tag{21}$$

where $\mathcal{L}_{\text{type}} = \sum_i \text{CE}(y_i^{\text{type}}, \hat{y}_i^{\text{type}})$ is the element type classification loss, computed via cross-entropy over predicted types, $\mathcal{L}_{\text{rel}} = \sum_{(i,j)} \text{BCE}(r_{ij}, \hat{r}_{ij})$ supervises pairwise relationships using binary cross-entropy, where r_{ij} denotes the ground-truth relation between elements i and j, $\mathcal{L}_{\text{coh}} = \|G - \hat{G}\|_2^2$ enforces global layout coherence, where G is the ground-truth graph embedding and $\hat{G}$ is the predicted one.

The navigation policy is trained using supervised learning:

$$\mathcal{L}_{\text{nav}} = -\sum_t \log p(a_t \mid s_t) \tag{22}$$

where a_t is the ground-truth action, p_t is the predicted probability distribution. The answer generation loss is defined as:

$$\mathcal{L}_{\text{ans}} = \sum_{l=1}^{L} -\log p(w_l | w_{<l}, s_{\text{EOA}}), \tag{23}$$

The total training loss integrates all three components:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{nav}} + \lambda \mathcal{L}_{\text{ans}} + \gamma \mathcal{L}_{\text{semantic}}, \tag{24}$$

where λ and γ are hyperparameters.

Experiments

Datasets and Evaluation Metrics

We evaluate our model primarily on WebVLN dataset (Chen et al. 2024), which jointly measures navigation success and QA accuracy. We further assess performance and efficiency on two widely used datasets for web manipulation tasks: Mind2Web (Deng et al. 2023) and WebArena (Zhou et al. 2023).

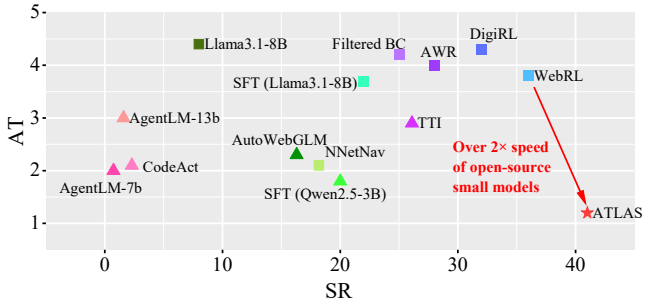

Figure 3: **Task success rate (SR) and Average Time Per Step (AT) on WebArena (Online).** ■: open-source models, ▲: closed-source models.

WebVLN contains 14,825 tasks that combine navigation with QA. Agents should reach the target page and answer questions based on its content. Evaluation includes navigation metrics—Success Rate (SR), Oracle Success Rate (OSR), Success Rate weighted by Path Length (SPL), and Trajectory Length (TL)—as well as QA metrics based on Wu-Palmer Similarity (WUPS) at thresholds 0.0 and 0.9.

Mind2Web includes 2,350 tasks from 137 websites across 31 domains, designed to test generalization across tasks, websites, and domains. The main metric is task Success Rate (SR), requiring all steps to be completed in the correct order.

WebArena consists of 812 tasks covering scenarios such as search, navigation, and manipulation. Evaluation focuses on task Success Rate (SR) and inference speed, measured by Average Time per step (AT) during online execution.

Implementation Details

The feature extraction backbone for webpage is a ResNet-152 model pre-trained on ImageNet (Anderson et al. 2018). The transformer layers for the NodeEncoder, ActionDecoder, QAEncoder, and QADecoder are set to 9, 2, 4, and 4 layers, respectively. During initialization, we adopt the OSCAR pre-trained weights to initialize the BERT model (Hong et al. 2020). For hyperparameter settings, we follow the same configuration as WebVLNNet, with $\alpha = 1$, $\lambda = 1$ and $\gamma = 1$ (Chen et al. 2024). Model training and inference are conducted using NVIDIA Tesla A100 and NVIDIA RTX 3090 GPUs. For the Mind2Web and WebArena datasets, we made task-specific modifications due to differences in action space compared to the WebVLN task.

Comparison with SOTAs

Web Navigation We compare our method with two categories of sota approaches: one category comprises web navigation-based methods (see Table 1), and the other includes LLM-based zero-shot/few-shot methods (see Table 2). Experimental results demonstrate that: (1) **ATLAS significantly improves web navigation performance**: On the validation set, ATLAS achieves a 12.5% ↑ in SR, a 16.56% ↑

Methods	Validation						Test					
	SR↑	OSR↑	SPL↑	TL↓	WUPS0.9↑	WUPS0.0↑	SR↑	OSR↑	SPL↑	TL↓	WUPS0.9↑	WUPS0.0↑
Random	0.05	0.17	0.02	6.49	0.00	0.00	0.04	0.17	0.02	6.50	0.00	0.00
VLNBERT (Hong et al. 2020)	17.59	17.59	16.73	6.81	9.99	13.91	11.28	12.04	10.75	7.55	7.12	9.26
VLNBERT*	18.62	18.62	18.14	6.96	11.23	14.98	12.23	12.23	11.74	7.72	8.50	10.36
WebGUM (Furuta et al. 2023)	6.02	6.02	6.02	**2.99**	1.84	4.08	9.71	9.71	9.71	**3.15**	3.57	6.98
WebGUM†	31.22	31.78	31.22	3.44	18.26	24.88	29.29	29.39	29.26	3.44	17.34	23.48
WebVLN-Net (Chen et al. 2024)	39.46	39.54	39.46	3.71	24.26	31.87	34.76	34.80	34.59	4.34	22.13	28.58
ATLAS (Ours)	**51.51**	**56.10**	**51.44**	3.90	**31.21**	**41.43**	**51.79**	**56.72**	**51.73**	4.48	**30.56**	**41.54**

Table 1: **Comparison with Navigation-Based Methods on WebVLN Dataset.** * denotes the model is initialized by LXMERT. WebGUM and WebGUM† denote the models based on T5-small and T5-base separately.

Methods	SR ↑	OSR ↑	SPL ↑	TL ↓	WUPS0.9 ↑	WUPS0.0 ↑
AgentBench [1]	6.97	11.94	4.13	5.39	1.64	4.65
NavGPT [2]	7.46	12.94	4.53	4.98	2.43	5.23
NavGPT*	16.92	21.89	11.61	5.43	5.97	12.06
ActionVerse [3]	40.02	41.18	39.78	5.62	-	-
ATLAS (Ours)	**51.79**	**56.72**	**51.73**	**4.48**	**30.56**	**41.54**

Table 2: **Comparison with LLM-Based Methods on WebVLN Dataset.** * denotes using GPT4. Research works: [1]: (Liu et al. 2023b), [2]: (Zhou, Hong, and Wu 2024) [3]: (Wang, Zhuang, and Wu 2025).

Method	Cross-Task	Cross-Website	Cross-Domain	Avg
GPT-3.5-Turbo	17.4	16.2	18.6	17.4
GPT-4†	36.2	30.1	26.4	30.9
Flan-T5-XL-3B*	52.0	38.9	39.6	43.5
LLaMA2-7B*	52.7	47.1	50.3	50.1
LLaMA2-70B*	55.8	51.6	55.7	54.4
Qwen-VL*	12.6	10.1	8.0	10.2
SeeClick* (Cheng et al. 2024)	23.7	18.8	20.2	20.9
CogVLM (Wang et al. 2024)	37.1	23.4	26.3	23.9
AutoWebGLM (Lai et al. 2024)	66.4	56.4	55.8	59.5
CogAgent (Hong et al. 2024)	62.3	54.0	59.4	58.2
ATLAS (1.5B)	68.0	58.2	57.2	61.0

Table 3: **Task Success Rate on Mind2Web.** The best scores are highlighted in red and the secondary scores are highlighted in blue. Avg: Average. ∗ indicates the model's fine-tuning on training set, and † denotes element selection from top-10 element candidates, others from top-50, following previous works (Lai et al. 2024).

Method	Red.	Git.	CMS	Map	Shop.	Avg
Qwen2.5-3B	5.3	13.3	5.7	0.0	4.4	6.1
Llama3.1-8B	5.3	10.0	5.7	15.4	8.9	8.5
Qwen2.5-32B	10.5	20.0	20.0	19.2	17.8	16.9
GPT-4o	10.5	20.0	20.0	20.0	11.1	13.9
GPT-4o-Turbo	10.5	16.7	14.3	36.7	13.3	17.6
QwQ-32B	15.8	33.3	25.7	15.4	20.0	22.4
OpenAI-o3	36.8	46.7	45.7	38.5	33.3	39.4
OpenAI-o4-mini	47.4	43.3	45.7	26.9	28.9	36.9
SteP* (Bai et al. 2024)	59.4	31.7	24.2	30.3	36.9	33.5
LM-TS* (Koh et al. 2024)	10.5	13.3	16.5	25.8	28.1	19.2
BC† (Torabi, Warnell, and Stone 2018)	36.8	6.7	20.0	33.3	17.8	20.6
AWR† (Peng et al. 2019)	57.9	26.7	31.4	26.7	17.8	28.5
Filtered BC† (Pan et al. 2024)	52.6	20.0	31.4	23.3	8.9	23.0
DigiRL† (Bai et al. 2024)	57.9	26.7	37.1	33.3	17.8	30.3
ATLAS (Ours)	52.0	45.0	50.0	32.0	36.0	41.0

Table 4: **Task success rate on WebArena-Lite (Offline).** The best scores are highlighted in red and the secondary scores are highlighted in blue. Avg: Average. ∗: model based on GPT-4o. †: model based on Llama3.1-8B.

ATL	SM	GP	F	SR ↑	OSR ↑	SPL ↑	TL ↓	WUPS0.9 ↑	WUPS0.0 ↑
	✓	✓	✓	30.98	37.88	30.99	**3.32**	18.54	24.88
✓		✓	✓	39.55	47.87	39.56	4.92	27.52	33.71
✓	✓		✓	40.65	47.48	40.52	4.66	28.55	34.37
✓	✓	✓		40.66	47.48	40.45	4.61	38.55	34.72
✓	✓	✓	✓	**51.79**	**56.72**	**51.73**	4.48	**30.56**	**41.54**

Table 5: **Ablation Study on System Architecture.**

in OSR, and an 11.98 ↑ in SPL compared to the navigation-based sota method WebVLN-Net. The performance gains on the test set are even more substantial, with a 17.03% ↑ in SR, a 21.92% ↑ in OSR, and a 17.14 ↑ in SPL. Compared to LLM-based approaches, ATLAS not only achieves higher navigation SR but also achieves the lowest TL, reducing it by 10% relative to sota. This indicates that ATLAS makes fewer erroneous page jumps or suboptimal search decisions, further validating the effectiveness of adaptive graph construction. (2) **ATLAS substantially improves QA performance**: Compared to WebVLN-Net, ATLAS achieves a 6.95 ↑ in WUPS0.9 and a 9.56 ↑ in WUPS0.0 on the validation set. On the test set, the improvements are even more pronounced, with an 8.43 ↑ in WUPS0.9 and a 13.03 ↑ in WUPS0.0. These results indicate that ATLAS can not only accurately reach the target webpage but also effectively integrate webpage information to answer questions. (3) **ATLAS leads to an increase in trajectory length**: While ATLAS maintains the shortest TL compared to LLM-based methods, it exhibits a moderate increase in TL relative to navigation-based methods. specifically, a 5.12% ↑ over WebVLN-Net. This is primarily due to the graph construction process, which involves exploration, the agent needs sufficient exploration to identify relevant paths, particularly in dynamic web environments.

Web Manipulation Web manipulation tasks require agents to perform concrete operations on webpages, such as filling out forms, uploading files, and more. The results on Mind2Web and WebArena are presented in Table 3 and Table 4, respectively. Experimental findings demonstrate the following: (1) **ATLAS exhibits strong generalization across tasks and websites**: On both the Cross-Task and Cross-Website settings of Mind2Web, ATLAS outperforms sota methods. It also achieves higher SR than sota

$\mathcal{L}_{nav}$	$\mathcal{L}_{type}$	$\mathcal{L}_{rel}$	$\mathcal{L}_{coh}$	$\mathcal{L}_{ans}$	SR ↑	OSR ↑	SPL ↑	TL ↓	WUPS0.9 ↑	WUPS0.0 ↑
✓					47.78	51.45	43.85	4.66	9.75	10.30
✓	✓				47.85	52.08	47.82	4.65	10.04	11.85
✓	✓	✓			49.90	53.53	48.66	4.55	10.25	12.74
✓	✓	✓	✓		51.50	56.10	51.44	4.50	10.54	13.35
✓	✓	✓	✓	✓	**51.79**	**56.72**	**51.73**	**4.48**	**30.56**	**41.54**

Table 6: **Ablation Study on Training Loss.**

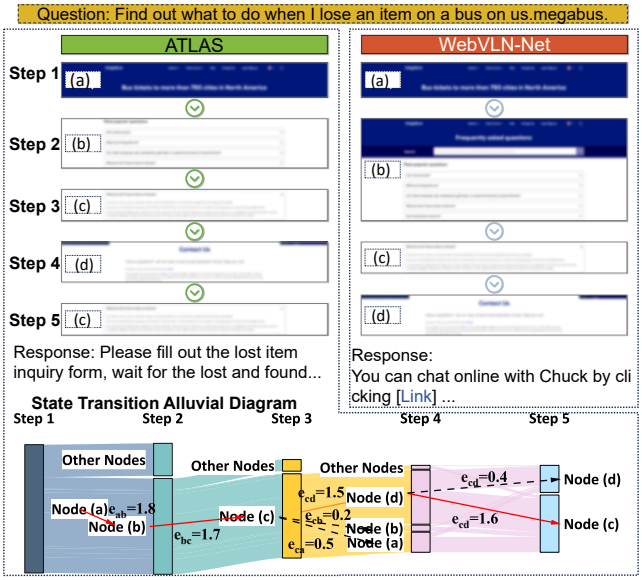

Figure 4: **Qualitative Analysis of ATLAS and WebVLN-Net.** In this task, ATLAS successfully guided the user to fill out the lost and found form and wait for further processing, whereas WebVLN-Net provided relatively irrelevant suggestions, such as advising the user to contact customer service via online chat or email.

baselines on both datasets, with 1.5% ↑ on Mind2Web and 1.6% ↑on WebArena. (2) **ATLAS still faces challenges in information integration and tool usage**: In environments such as Reddit, where successful task completion requires reasoning over posts, replies, and search content, and Map, which involves interacting with multiple software tools, ATLAS performs relatively worse—without major modifications to its system components. Nevertheless, ATLAS still demonstrates strong performance on long-horizon tasks, achieving 4.3% ↑ in the CMS environment. (3) **ATLAS strikes a balance between efficiency and performance**: Figure 3 compares the SR and AT of different models under online evaluation. Compared with the proprietary LLM pipeline (ScribeAgent + GPT-4o), ATLAS achieves over 10× speedup with a 10% ↓ in SR. Moreover, while being more than 2× faster, ATLAS consistently outperforms open-source lightweight LLM-based baselines.

Qualitative Analysis We present a case study comparing ATLAS and WebVLN-Net, as illustrated in Figure 4. After correctly navigating to the target page, ATLAS further explored subsequent pages (the "Contact Us" page), but upon identifying that the additional content was less relevant to the question, it effectively backtracked and produced an accurate answer. In contrast, WebVLN-Net performed a linear navigation through the webpages and, although it generated a plausible answer, its relevance to the question was relatively weak. This demonstrates that the graph-based strategy adopted by ATLAS enables a more effective balance between exploration and exploitation, resulting in more accurate web navigation and question answering.

Ablation Study

We design two types of ablation studies to evaluate the effectiveness of the system architecture and training strategy.

System Architecture: We conducted ablation studies by individually removing four key components of the ATLAS architecture: Adaptive Topological Layout Construction (ATL), Semantic Mapping module (SM), Graph-Guided Navigation Planning (GP), and Feature Fusion (F), as shown in Table 5. The results demonstrate the following: (1) **The inclusion of ATL significantly enhances system performance**: Removing ATL leads to a substantial drop in SR from 51.79 to 30.98 and in SPL from 51.73 to 30.99, highlighting the critical role of adaptive layout construction in effective web navigation. (2) **Each module plays a vital role in the overall system**: These components contribute complementarily by bridging web structural understanding with semantic interpretation. The integration of all modules enables ATLAS to effectively balance navigation efficiency and answer accuracy.

Training Strategy: We conduct an ablation study on the loss function, as shown in Table 6. The results demonstrate that the strategy incorporating all five loss components achieves the best performance across nearly all evaluation metrics. Specifically, adding $\mathcal{L}_{rel}$ and $\mathcal{L}_{coh}$ on top of $\mathcal{L}_{nav}$ and $\mathcal{L}_{type}$ significantly improves navigation performance (SR increases from 47.85 to 51.50) and enhances semantic relevance in answers (WUPS@0.0 improves from 11.85 to 13.35). Introducing $\mathcal{L}_{ans}$ leads to a substantial gain in answer accuracy, with WUPS@0.0 jumping from 13.35 to 41.54. These findings underscore the importance of jointly optimizing all loss terms for achieving robust performance in both navigation and question answering tasks.

Conclusion

In this paper, we propose ATLAS: Adaptive Topological Layout And Semantic mapping, a framework that dynamically constructs webpage topological relations through an adaptive mechanism and aligns the distribution of web elements via semantic mapping. This enables the integration of global planning and local decision-making for effective web navigation and QA. ATLAS achieves over a 10% improvement in success rate compared to sota methods on the WebVLN dataset. It also improves task success rates by 1.5% and 1.6% on the Mind2Web and WebArena datasets, respectively, while delivering a 2–5× speedup over existing open-source LLM-based approaches. Although ATLAS exhibits certain limitations in information integration and tool usage, it offers a lightweight solution for the design of web agents.

Acknowledgments

This research was partly supported by grants of National Natural Science Foundation of China (NSFC, Grant No. 62171281), Science and Technology Commission of Shanghai Municipality (STCSM, Grant Nos. 25GA32001003).

References

Anderson, P.; Wu, Q.; Teney, D.; Bruce, J.; Johnson, M.; Sünderhauf, N.; Reid, I.; Gould, S.; and Van Den Hengel, A. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3674–3683.

Bai, H.; Zhou, Y.; Pan, J.; Cemri, M.; Suhr, A.; Levine, S.; and Kumar, A. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 12461–12495.

Chang, Y.; Narang, M.; Suzuki, H.; Cao, G.; Gao, J.; and Bisk, Y. 2022. Webqa: Multihop and multimodal qa. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16495–16504.

Chen, Q.; Pitawela, D.; Zhao, C.; Zhou, G.; Chen, H.-T.; and Wu, Q. 2024. Webvln: Vision-and-language navigation on websites. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 1165–1173.

Cheng, K.; Sun, Q.; Chu, Y.; Xu, F.; Li, Y.; Zhang, J.; and Wu, Z. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113.

Deng, X.; Gu, Y.; Zheng, B.; Chen, S.; Stevens, S.; Wang, B.; Sun, H.; and Su, Y. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36: 28091–28114.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Durrant-Whyte, H.; and Bailey, T. 2006. Simultaneous localization and mapping: part I. *IEEE robotics & automation magazine*, 13(2): 99–110.

Furuta, H.; Lee, K.-H.; Nachum, O.; Matsuo, Y.; Faust, A.; Gu, S. S.; and Gur, I. 2023. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arXiv:2305.11854*.

Gur, I.; Furuta, H.; Huang, A.; Safdari, M.; Matsuo, Y.; Eck, D.; and Faust, A. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*.

He, H.; Yao, W.; Ma, K.; Yu, W.; Dai, Y.; Zhang, H.; Lan, Z.; and Yu, D. 2024. WebVoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

He, K.; Huang, Y.; Wu, Q.; Yang, J.; An, D.; Sima, S.; and Wang, L. 2021. Landmark-rxr: Solving vision-and-language navigation with fine-grained alignment supervision. *Advances in Neural Information Processing Systems*, 34: 652–663.

Hong, W.; Wang, W.; Lv, Q.; Xu, J.; Yu, W.; Ji, J.; Wang, Y.; Wang, Z.; Dong, Y.; Ding, M.; et al. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14281–14290.

Hong, Y.; Wu, Q.; Qi, Y.; Rodriguez-Opazo, C.; and Gould, S. 2020. A recurrent vision-and-language bert for navigation. *arXiv preprint arXiv:2011.13922*.

Jia, S.; Kiros, J.; and Ba, J. 2019. Dom-q-net: Grounded rl on structured language. *arXiv preprint arXiv:1902.07257*.

Jiang, J.; Zhou, K.; Dong, Z.; Ye, K.; Zhao, W. X.; and Wen, J.-R. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1–2): 99–134.

Kim, G.; Baldi, P.; and McAleer, S. 2023. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36: 39648–39677.

Koh, J. Y.; McAleer, S.; Fried, D.; and Salakhutdinov, R. 2024. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*.

Lai, H.; Liu, X.; Iong, I. L.; Yao, S.; Chen, Y.; Shen, P.; Yu, H.; Zhang, H.; Zhang, X.; Dong, Y.; et al. 2024. AutoWebGLM: A Large Language Model-based Web Navigating Agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5295–5306.

Liu, E. Z.; Guu, K.; Pasupat, P.; Shi, T.; and Liang, P. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*.

Liu, X.; Lai, H.; Yu, H.; Xu, Y.; Zeng, A.; Du, Z.; Zhang, P.; Dong, Y.; and Tang, J. 2023a. WebGLM: towards an efficient web-enhanced question answering system with human preferences. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, 4549–4560.

Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; et al. 2023b. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Liu, Z.; Yao, W.; Zhang, J.; Xue, L.; Heinecke, S.; Murthy, R.; Feng, Y.; Chen, Z.; Niebles, J. C.; Arpit, D.; et al. 2023c. Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents. *arXiv preprint arXiv:2308.05960*.

Nakano, R.; Hilton, J.; Balaji, S.; Wu, J.; Ouyang, L.; Kim, C.; Hesse, C.; Jain, S.; Kosaraju, V.; Saunders, W.; et al.

2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Pan, J.; Zhang, Y.; Tomlin, N.; Zhou, Y.; Levine, S.; and Suhr, A. 2024. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*.

Peng, X. B.; Kumar, A.; Zhang, G.; and Levine, S. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Shaw, P.; Joshi, M.; Cohan, J.; Berant, J.; Pasupat, P.; Hu, H.; Khandelwal, U.; Lee, K.; and Toutanova, K. N. 2023. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. *Advances in Neural Information Processing Systems*, 36: 34354–34370.

Shi, T.; Karpathy, A.; Fan, L.; Hernandez, J.; and Liang, P. 2017. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, 3135–3144. PMLR.

Torabi, F.; Warnell, G.; and Stone, P. 2018. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, W.; Lv, Q.; Yu, W.; Hong, W.; Qi, J.; Wang, Y.; Ji, J.; Yang, Z.; Zhao, L.; XiXuan, S.; et al. 2024. Cogvlm: Visual expert for pretrained language models. *Advances in Neural Information Processing Systems*, 37: 121475–121499.

Wang, X.; Zhuang, B.; and Wu, Q. 2025. Action as a Modality: Turning Multi-Modal LLMs to General Action Planners. *OpenReview*. Available at https://openreview.net/pdf?id=jaIxmAVAqF.

Xu, Y.; Lu, D.; Shen, Z.; Wang, J.; Wang, Z.; Mao, Y.; Xiong, C.; and Yu, T. 2024. AgentTrek: Agent Trajectory Synthesis via Guiding Replay with Web Tutorials. *arXiv preprint arXiv:2412.09605*.

Xue, T.; Qi, W.; Shi, T.; Song, C. H.; Gou, B.; Song, D.; Sun, H.; and Su, Y. 2025. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*.

Yang, Z.; Li, L.; Lin, K.; Wang, J.; Lin, C.-C.; Liu, Z.; and Wang, L. 2023. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9(1): 1.

Yao, S.; Chen, H.; Yang, J.; and Narasimhan, K. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35: 20744–20757.

Zeng, A.; Liu, M.; Lu, R.; Wang, B.; Liu, X.; Dong, Y.; and Tang, J. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

Zheng, B.; Gou, B.; Kil, J.; Sun, H.; and Su, Y. 2024. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*.

Zhou, G.; Hong, Y.; and Wu, Q. 2024. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 7641–7649.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.