Auto-Intent: Automated Intent Discovery and Self-Exploration for Large Language Model Web Agents

Jaekyeom Kim[1] **Dong-Ki Kim**[2] **Lajanugen Logeswaran**[1]
Sungryull Sohn[1] **Honglak Lee**[1,3]

[1]LG AI Research [2]Field AI [3]University of Michigan

[1] {jaekyeom, llajan, srsohn, honglak}@lgresearch.ai [2] dongkikim93@gmail.com

Abstract

In this paper, we introduce *Auto-Intent*, a method to adapt a pre-trained large language model (LLM) as an agent for a target domain without direct fine-tuning, where we empirically focus on web navigation tasks. Our approach first discovers the underlying intents from target domain demonstrations unsupervisedly, in a highly compact form (up to three words). With the extracted intents, we train our *intent predictor* to predict the next intent given the agent's past observations and actions. In particular, we propose a *self-exploration* approach where top-k probable intent predictions are provided as a hint to the pre-trained LLM agent, which leads to enhanced decision-making capabilities. Auto-Intent substantially improves the performance of GPT-{3.5, 4} and Llama-3.1-{70B, 405B} agents on the large-scale real-website navigation benchmarks from Mind2Web and online navigation tasks from WebArena with its cross-benchmark generalization from Mind2Web.

1 Introduction

Recently, large language models (LLMs) pre-trained on a massive amount of data (Achiam et al., 2023; Dubey et al., 2024) have excelled at reasoning and a variety of tasks. They exhibit robust planning and reasoning abilities, enabling LLM agents to perform diverse tasks (Wang et al., 2023; Xi et al., 2023; Zeng et al., 2023). However, these agents often face challenges in domains with less prior knowledge, especially ones with large action spaces, such as navigating websites or operating mobile devices (Cheng et al., 2024; Hong et al., 2023; Koh et al., 2024).

We explore improving decision-making with pre-trained LLMs on downstream tasks by injecting domain knowledge into the input context, in the form of natural language hints for the next action. This allows them to fully retain their strong general reasoning capabilities while avoiding overly costly or impossible fine-tuning. Leveraging natural language guidance for improving LLM planning and reasoning capabilities has found much success in prior work (Wei et al., 2022; Yao et al., 2022; Shinn et al., 2024; Fu et al., 2024; Zhao et al., 2024).

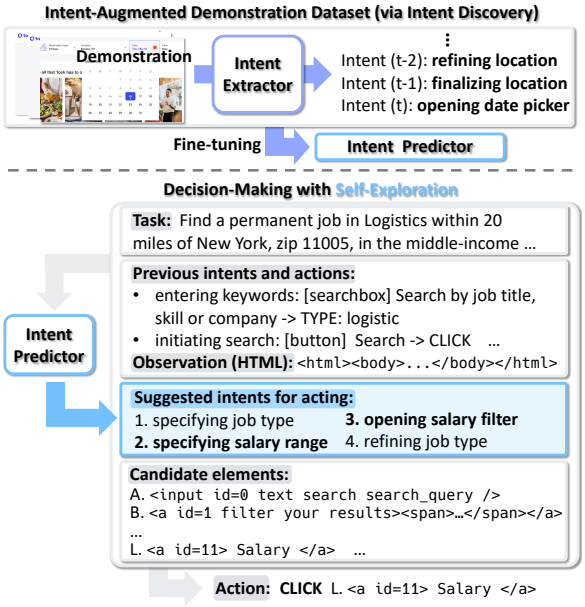

Figure 1: **Overview of *Auto-Intent*:** Given a dataset of demonstration trajectories, we first extract natural language *intents* in an unsupervised manner and train an intent predictor. Enforcing the intents to be concise phrases and providing top-k intent predictions as hints to an LLM agent allows efficient internal exploration of semantically diverse intent hypotheses, resulting in improved action prediction. See text for details.

Although prior work has shown that LLMs have strong priors to reason about intermediate subgoals (Logeswaran et al., 2022; Huang et al., 2022; Hao et al., 2023), the resulting performance can be largely affected by the injected hints' accuracy, which could be limited especially in complex environments such as real-world web navigation with numerous elements and possible actions. In this work, we aim to improve the LLM agent's performance further by proposing *self-exploration*. Our

Findings of the Association for Computational Linguistics: EACL 2024, pages 16531–16541
November 12-16, 2024

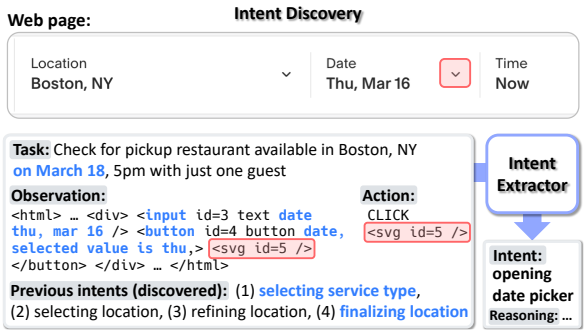

Figure 2: A hard example of intent discovery: the action (`CLICK <svg id=5 />`) does not provide any semantics about the intent. Our intent extractor successfully discovers the *underlying* intent by thoroughly understanding the context and connecting to the relevant parts.

key insight is to provide multiple plausible and semantically varied hints that we call *intents* to the LLM agent for flexible reasoning and acting given a set of possible directions. To achieve this, we constrain intents to very short phrases and generate top-k intents to provide as a collective hint with a beam search using a smaller model fine-tuned for intent prediction. This fine-tuning is enabled by discovering intents from demonstration data with our *intent extractor*. The compact intent space encourages semantically distinct intents to be sampled (as opposed to syntactically diverse intents that are semantically identical). This *self-exploration* with multiple intents enhances the agent to find the correct directions and associated actions. See Figure 1 for an illustration of our approach.

Our main contributions are as follows:

- We introduce *Auto-Intent*, a method to extract natural language intents from demonstration trajectories in an unsupervised manner and leverage intents as hints for pre-trained LLM agents through a fine-tuned intent prediction model.
- We present a *self-exploration* strategy where the LLM agent reviews varied plausible intents suggested by the intent prediction model and demonstrate that this results in more accurate action prediction compared to relying on a single intent.
- We empirically show that the injection of predicted top-k intents effectively improves the performance of GPT-{3.5, 4} and Llama-3.1{-70B, 405B} agents on the large-scale real-website benchmark tasks from Mind2Web (Deng et al., 2024) and online navigation tasks from WebArena (Zhou et al., 2023) in a cross-benchmark generalization setting from Mind2Web.

2 Auto-Intent: Intent Discovery and Self-Exploration with Intent Prediction

To address the inadequate domain knowledge in pre-trained LLM agents, we introduce an abstract natural language representation we refer to as an *intent*, which hints what the agent can perform next. We aim to enhance LLM agents further without limiting them by the intent prediction model's performance, via providing top-k predicted intents as a set of probable directions to consider. We describe the problem definition (Section 2.1), design of the intent space and discovery of underlying intents from demonstrations in an unsupervised manner (Section 2.2), and fine-tuning and using the intent prediction model for acting with top-k probable intents as a flexible hint (Section 2.3) in detail.

2.1 Problem Statement

We consider sequential decision-making for completing each given task. At each time step t starting from $t = 1$, the agent receives an observation $\boldsymbol{o}_t \in \mathcal{O}$ and performs an action $\boldsymbol{a}_t \in \mathcal{A}$ until the episode ends, with access to previous observations and actions. We use a demonstration dataset $\mathcal{D}_{\text{demo}} = \{\boldsymbol{\tau}_i\}_{i=1}^{N}$, where each trajectory $\tau = \{(\boldsymbol{o}_t, \boldsymbol{a}_t)\}_{t=1}^{T}$ consists of observations and actions from the same episode. Empirically, we put our focus on real-world web navigation tasks.

2.2 Intent Space and Discovery

Intent space design. We aim to provide a semantically varied set of predicted intents to be examined by the LLM policy for more flexible reasoning and improved action prediction. Given a vocabulary V, we define our intent space as $\mathcal{Z} = V^L$ where L is a small number. We find expressing each intent using only up to $L = 3$ words in the form *gerund + noun phrase (object)* appropriate for our use with the desired expressiveness while being computationally efficient. Thanks to its compactness, even single-word changes can lead to clear semantic distinctions (*e.g.*, *selecting date* vs. *selecting time* vs. *selecting guests*). The smaller semantic overlap between different intents makes the intent space suitable for specifying more varied directions using the same number of intents, which fits our goal.

Intent discovery. With the intent space $\mathcal{Z}$, we define the intent discovery procedure with a prompt-

based intent extractor $\mathcal{M}_{\texttt{extract}}$ as

$$\boldsymbol{z}_t = \mathcal{M}_{\texttt{extract}}(\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{z}_{1:t-1}) \quad (1)$$

where $\boldsymbol{z}_t \in \mathcal{Z}$ denotes the intent discovered for time step t. We instruct it to take the observation (including task description), action, and previous-step intents together into account to discover the intent. Refer to Figure 2 for a hard example that requires a contextual understanding and Appendix A.3 for our full prompt.

Intent-augmented demonstrations. Given the dataset $\mathcal{D}_{\texttt{demo}}$, we discover intents using Equation (1) for each step. We construct an intent-augmented demonstration set $\mathcal{D}_{\texttt{intent}} = \{\boldsymbol{\tau}'_i\}_{i=1}^{N}$ where each trajectory is $\boldsymbol{\tau}' = \{(\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{z}_t)\}_{t=1}^{T}$.

2.3 Self-Exploration with Intent Prediction and Acting with LLMs

Intent predictor. Using the intent-augmented demonstration dataset $\mathcal{D}_{\texttt{intent}}$ from Section 2.2, we train a smaller language model to predict each discovered natural language intent $\boldsymbol{z}_t$ given $\boldsymbol{o}_t, \boldsymbol{a}_{1:t-1}, \boldsymbol{z}_{1:t-1}$ as input. We employ this model trained on $\mathcal{D}_{\texttt{intent}}$ as our intent predictor, $\mathcal{M}_{\texttt{intent}}$. See Appendix A.4 for the training details.

Intent prediction. One important property of the intents that $\mathcal{M}_{\texttt{intent}}$ outputs is the compactness. Thanks to the definition of our compact intent space $\mathcal{Z}$ with a small L from Section 2.2, multiple intent predictions can span a broader spectrum of meanings and thus improve the recall of the correct intent effectively. Therefore, we employ the generation of multiple intent predictions with $\mathcal{M}_{\texttt{intent}}$ for finding the correct intent, which is expressed as

$$\hat{\boldsymbol{z}}_t^1, \ldots, \hat{\boldsymbol{z}}_t^k \sim \mathcal{M}_{\texttt{intent}}(\boldsymbol{o}_t, \boldsymbol{a}_{1:t-1}, \boldsymbol{z}_{1:t-1}) \quad (2)$$

where the previous intents $\boldsymbol{z}_{1:t-1}$ are obtained with $\mathcal{M}_{\texttt{extract}}$ using Equation (1). The generated top-k intents can be employed as a set of probable, different directions for the LLM policy, providing the ingredients for *self-exploration*. While different generation strategies might be applicable depending on the requirements (*e.g.*, more semantic diversities of the intents), we find beam search effective and efficient enough for our top-k intent prediction.

LLM policy with self-exploration. We incorporate the top-k intents $\hat{\boldsymbol{z}}_t^{1:k}$ as a concatenated list into the input prompt for the LLM policy π:

$$\boldsymbol{a}_t = \pi(\boldsymbol{o}_{1:t}, \boldsymbol{a}_{1:t-1}, \hat{\boldsymbol{z}}_t^{1:k}). \quad (3)$$

We instruct the LLM to examine the suggested intents together to act with an appropriate one. Combined with the intent prediction, the agent internally infers top-k intents and reasons with them as a set of probable directions for acting, which we refer to as *self-exploration*. Its exploration effect is achieved implicitly and internally, unlike exploration via environment interactions in reinforcement learning. This can be especially effective in complex environments where predicting the correct intent on the first try is challenging. See Appendix A.5 for the prompt.

3 Experiments

3.1 Setup for Main Evaluation

Evaluation. We evaluate our approach on a large-scale real website navigation dataset, Mind2Web (Deng et al., 2024). Its three test splits evaluate agents' generalization to unseen (a) tasks, (b) websites, and (c) domains. "Elem. acc" measures the accuracy with respect to the correct elements, "Op, F1" is a metric based on string matching, and "Step SR" denotes the rate of *fully successful* steps. Refer to Appendix A.1 for more details.

Compared methods. We compare our results with MindAct (Deng et al., 2024), a directly trained agent with the same backbones, and SeeAct (Zheng et al., 2024), a prompt-based agent with GPT-4V. For all method, we use the same pre-processing of keeping only top-N candidate elements by Deng et al. (2024). We examine Flan-T5$_{\text{XL}}$ and Mistral-7B as both MindAct baselines and our intent predictor. Refer to Appendix A for more details.

3.2 Main Evaluation Results

Table 1 presents our main evaluation results on Mind2Web. Our method significantly enhances not only the GPT-3.5 agent but also the much stronger GPT-4, Llama-3.1-70B, and Llama-3.1-405B-FP8 agents in all cases with both Flan-T5$_{\text{XL}}$ and Mistral-7B intent predictors, which suggests its effectiveness. Overall, it brings noteworthy improvements to the element accuracies, which thus contribute to the step success rates as well. Our intent predictor fine-tuned on the train set produces larger improvements on the cross-task split, but we observe its efficacy even on the more challenging generalization splits, cross-website and cross-domain, outperforming MindAct with the same backbones and SeeAct as well.

Methods	Cross-task			Cross-website			Cross-domain		
	Elem. acc	Op. F1	Step SR	Elem. acc	Op. F1	Step SR	Elem. acc	Op. F1	Step SR
MindAct (Flan-T5$_{XL}$, 3B)	55.1	75.7	**52.0**	42.0	65.2	38.9	42.1	66.5	39.6
MindAct (Mistral-7B†)	53.7	76.8	50.1	41.7	67.0	38.1	43.5	67.8	40.3
SeeAct (GPT-4V)	46.4	73.4	40.2	38.0	67.8	32.4	42.4	69.3	36.8
ICL (GPT-3.5)	30.5	67.5	27.2	24.9	59.5	22.7	29.8	62.7	27.3
w/ Auto-Intent (Flan-T5$_{XL}$, 3B)	44.1	71.9	38.8	37.1	62.6	30.7	38.9	64.8	35.0
w/ Auto-Intent (Mistral-7B†)	42.9	71.1	37.3	36.0	61.3	29.5	37.8	63.9	34.2
ICL (GPT-4)	47.5	69.9	41.5	44.6	64.2	38.4	44.4	65.7	40.2
w/ Auto-Intent (Flan-T5$_{XL}$, 3B)	55.8	73.3	50.1	47.6	64.0	40.0	47.3	66.3	42.5
w/ Auto-Intent (Mistral-7B†)	53.8	71.8	47.6	48.6	63.9	41.2	46.9	65.9	42.3
ICL (GPT-4)*	46.9	75.2	41.7	45.0	**70.9**	40.0	45.3	72.3	41.3
/w Auto-Intent (Mistral-7B†)*	53.3	**77.0**	47.3	49.3	69.9	42.0	48.8	72.3	44.1
ICL (Llama-3.1-70B)*	43.9	68.9	37.3	40.8	63.6	34.0	42.6	66.5	37.0
/w Auto-Intent (Mistral-7B†)*	51.2	75.3	44.6	44.4	67.2	36.9	46.8	70.4	41.5
ICL (Llama-3.1-405B-FP8)*	50.4	74.2	43.6	46.8	67.5	39.9	47.1	70.7	41.6
/w Auto-Intent (Mistral-7B†)*	**56.3**	76.9	50.4	**51.1**	70.1	**43.6**	**49.5**	**72.5**	**44.6**

Table 1: Performance comparison on Mind2Web (Deng et al., 2024). MindAct (Flan-T5$_{XL}$, 3B) (Deng et al., 2024) and SeeAct (Zheng et al., 2024) results are from their papers. † denotes LoRA (Hu et al., 2021) fine-tuning. Our in-context learning (ICL) runs use top-20 candidate elements except for ones with *, which use top-40 candidates.

Methods	Task success rate
ICL (GPT-4)	19.0%
/w Auto-Intent (Mistral-7B†)	23.8%
ICL (Llama-3.1-405B-FP8)	14.3%
/w Auto-Intent (Mistral-7B†)	19.0%

Table 2: Online evaluation of our agent on a subset of the Shopping split of WebArena (Zhou et al., 2023). Our intent predictors are trained on and transferred from Mind2Web. † denotes LoRA fine-tuning.

Methods	Elem. acc	Op. F1	Step SR
GPT-4 w/o intents	54.3	79.0	47.9
GPT-4 w/ 1 discovered intent	**73.8**	**83.7**	**64.0**

Table 3: Performance comparison of the GPT-4 baseline agent without intents (row 1) and the GPT-4 agent with a single intent discovered with our intent extractor as an injected hint (row 2), on 50 randomly selected tasks from the train split of Mind2Web (Deng et al., 2024).

3.3 Online Evaluation Results with Cross-Benchmark Generalization

To evaluate Auto-Intent in an online setting where agents need to interact with live websites, we conduct experiments on tasks from WebArena (Zhou et al., 2023) to leverage the automatic evaluators they provide. Specifically, we employ our intent predictors trained on the train split of Mind2Web as-is for this online evaluation in the WebArena environment, which allows us to examine Auto-Intent in two aspects: its applicability to online environments and generalization capabilities.

Table 2 shows the results of the online evaluation. Interestingly, in this cross-benchmark online evaluation, we find that our intent predictors trained on Mind2Web improve the performance of both GPT-4 and Llama-3.1-405B agents on Shopping tasks from WebArena. It suggests that not only can Auto-Intent be useful for enhancing the decision-making capabilities of LLM agents in online environments as well, but it can also generalize to a different domain from where it is trained, which could be helpful in practical scenarios, such as demonstration data scarcity in the target domain. Refer to Appendix A.6 for more details.

3.4 Empirical Analysis and Ablation Study

Q1. *Does our intent extractor discover underlying intents effectively?*

We provide Figure 2 as a qualitative example of intent discovery from hard samples. While the action (`CLICK <svg id=5 />`) does not carry any semantic information about the underlying intent, our intent extractor successfully discovers it by understanding the context from the task, observation, and previous intents. It shows the intent extractor's capabilities of identifying intents from demonstrations with enough understanding of interactions.

Additionally, in Table 3, we compare the per-

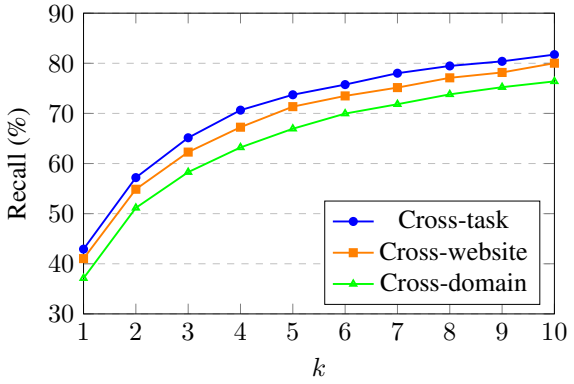

Figure 3: The intent label recalls with respect to top-k predicted intents on Mind2Web's test sets ($N = 20$).

formance of the GPT-4 baseline agent without any intents or hints and the agent with a single intent discovered with our intent extractor as a hint. It demonstrates that despite the conciseness of each discovered intent (up to three words), directly incorporating it into the LLM agent can bring significant performance improvements, which suggests the effectiveness of the intent extractor at discovering semantically valid intents from demonstrations.

Q2. *Is top-k intent prediction effective at finding the correct intent?*

We compare the top-k predicted intents with the intent labels discovered using ground-truth actions, on the three test splits. Figure 3 shows the average recalls of the intent labels with respect to the top-k predictions computed using sentence embedding similarities (see Appendix A.1 for details). We observe that the recall enhances as k increases, which suggests that the intent prediction provides the exploration effect for finding the appropriate intent in the intent space.

Q3. *Is self-exploration with top-k intents effective?*

We conduct an ablation study on self-exploration, where we compare Auto-Intent's performance with its variant that uses only the top-1 intent prediction without self-exploration. Table 4 shows the results on the random subset of the test splits. We find that on the cross-website and cross-domain test splits, where the generalization of the intent predictor $\mathcal{M}_{\texttt{intent}}$ is more challenging, only giving the top-1 predictions is considerably less efficacious than on the cross-task split and our self-exploration provides notable performance boosts.

Q4. *Is a top-k intent prediction an effective hint for correct action prediction?*

To examine how efficacious a top-k intent prediction hint is for predicting correct actions, we isolate the evaluation of intent hints from the LLM agent evaluation with those injected hints. For Table 4, we act with each of the top-k intents separately and aggregate the results to obtain the "Oracle select" performance with the *best* intent among the top-k. The significant improvement from the "GPT-4" and "Top-1" baselines suggests that the top-k intent prediction can be an effective hint for acting and employing a stronger pre-trained LLM might benefit the performance of our agent even further.

Methods	**Cross task**		**Cross website**		**Cross domain**	
	Ele. acc	Step SR	Ele. acc	Step SR	Ele. acc	Step SR
GPT-4	46.2	40.2	42.1	35.8	50.2	45.1
w/ Top-1 intent	53.2	46.0	43.6	37.9	52.5	46.2
w/ Auto-Intent	54.1	46.1	49.2	42.3	56.5	50.9
w/ Oracle select (top-5)	68.2	60.0	56.9	50.6	65.2	57.8

Table 4: Ablation with different intent injections on Mind2Web's random subset (50 tasks each, $N = 20$).

4 Conclusion

We investigated a way to improve LLM agents on downstream tasks where they possess insufficient domain knowledge. Our *Auto-Intent* discovers concise intents from demonstrations and predicts multiple, semantically varied intents so that our LLM policy examines the top-k intents for acting. On Mind2Web (Deng et al., 2024), a large-scale benchmark with real-website tasks, we empirically showed that our top-k intent prediction is effective for predicting correct actions and improving LLM agent's performance. In addition, we performed the evaluation of our approach in an online setting on Shopping tasks from the WebArena benchmark (Zhou et al., 2023), which suggests its applicability to online tasks and generalization capabilities to different domains from where it is trained.

Limitations

Our empirical investigation is limited to a web navigation setting. Although we choose Mind2Web (Deng et al., 2024) for our main evaluation as it provides a challenging, large-scale benchmark built based on many real websites and domains and different generalization problem settings, future work could examine the empirical effectiveness of our approach on more domains for decision-making, such as mobile device operation (Cheng et al., 2024).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024. Autoguide: Automated generation and selection of state-aware guidelines for large language model agents. *arXiv preprint arXiv:2403.08978*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2023. Cogagent: A visual language model for gui agents. *arXiv preprint arXiv:2312.08914*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*.

Lajanugen Logeswaran, Yao Fu, Moontae Lee, and Honglak Lee. 2022. Few-shot subgoal planning with language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Experimental Details

A.1 Dataset and Evaluation

Dataset We employ Mind2Web (license: CC BY 4.0, allows research purposes) (Deng et al., 2024), a large-scale web navigation dataset with task instructions and corresponding trajectories on 137 real websites. The dataset is in English and constructed by explicitly instructing annotators to refrain from using personal or sensitive information. The goal is to complete each given natural language task by performing a series of actions, where three types of actions exist: `CLICK`, `SELECT`, and `TYPE`. The agent needs to choose the target element to perform each action with, and each `SELECT` or `TYPE` action additionally requires a string value for selecting a specific option or typing the desired text, respectively. Mind2Web provides three test splits for evaluating web navigation agents' generalization capabilities. The cross-task split is the most in-distribution setting; it contains new tasks but in the domains and websites seen from the train split. The cross-website split has new tasks on unseen websites but in previously seen domains. Lastly, the cross-domain split is for testing with new tasks in unseen domains as well as websites. We summarize the information about the Mind2Web dataset and its statistics in Table 5.

Evaluation metrics. We employ the evaluation protocol by Deng et al. (2024). The element accuracy ("Elem. acc") measures whether the agent chose one of the ground-truth elements from the web page at each time step. The operation F1 ("Op. F1") is the F1 score for the predicted action (the type and string value) computed with respect to the ground-truth action. The step success rate ("Step SR") counts successful steps, where each step is considered successful only if the chosen target element is correct and the action type and string value match the ground truth. Following Deng et al. (2024), these three step-wise metrics are macro-averaged over tasks. For our empirical analysis based on embedding similarity (**Q2** from Section 3.4), we use `all-mpnet-base-v2` (Apache 2.0, allows research purposes) from SentenceTransformers (Reimers and Gurevych, 2019; Song et al., 2020).

A.2 Compared Methods

We employ the same element-ranking model suggested and provided by Deng et al. (2024). Given each element from the web page, the model outputs the score for its correctness as a target element. The element-ranking model alone is not as effective at predicting correct target elements by choosing the highest-scoring elements. However, as each web page often contains numerous candidate elements, the element-ranking model is used to reduce the set of candidate elements by keeping only top-N-scoring elements, as the first stage of the action prediction with all the compared methods. MindAct (Deng et al., 2024) uses $N = 50$ and conducts a tournament of elements, by grouping $N = 50$ candidate elements into sets of 5 or less. SeeAct (Zheng et al., 2024) groups $N = 50$ candidates into three batches and tries predicting the action given each with the screenshot. For our in-context learning (ICL) agents with or without intents, we predict the action in a single pass given all the top-N candidates at once.

Hyperparameter	Values
Attention	FlashAttention-2 (Dao, 2023)
LoRA rank	**64**, 128
LoRA α	**8**, 16
LoRA dropout rate	**0.1**
Label smoothing factor	**0.1**, 0
Learning rate	**5e−6**, 1e−6, 1e−5
Batch size	**64**
Epochs	**3**, 4

Table 6: Training hyperparameter search for our intent predictor with `Mistral-7B-v0.1` where the best values are bold-faced.

Hyperparameter	Values
Context length	**768**, 512
Label smoothing factor	**0.1**
Learning rate	**1e−5**, 1e−6, 5e−6
Batch size	**64**
Epochs	**3**

Table 7: Training hyperparameter search for our intent predictor with `Flan-T5-XL` where the best values are bold-faced.

Split	Domains	Websites	Tasks	Avg. horizon	Seen during training?		
					Tasks	Websites	Domains
Train	18	73	1,009	7.71	✓	✓	✓
Cross-task	18	69	252	8.31	✗	✓	✓
Cross-website	10	10	177	7.76	✗	✗	✓
Cross-domain	13	54	912	6.48	✗	✗	✗

Table 5: The statistics and information about Mind2Web (Deng et al., 2024), a large-scale web navigation dataset used for our evaluation.

Prompt for Intent Extractor

System

You are a helpful assistant that understands tasks in web environments and how to complete them step by step. Within the HTML webpage, given the final goal, previous step subgoals, and the current ground-truth action, your job is to describe what the current action is achieving in the current step, with respect to the final goal and previous step subgoals. The inferred current step subgoal must be a concise gerund (verb -ing) phrase that consists of two or three words, where the first word is the gerund and the second (and possibly third) word(s) is a noun (phrase) as the object but without any named entities or proper nouns. Describing the current step subgoal should be based on and connected to the final goal (and previous step subgoals). (For example, if the final goal only specifies "to find the highest-rated accommodation" but not its type, and the accommodation chosen by the current action just happens to be a hotel, the subgoal must not mention "hotel" at all because it is not part of the instruction from the final goal.) The target element of the current action as well as other elements appear in the given HTML. Try to describe the current step subgoal, so that the subgoal combined with the final goal can uniquely identify the target element among all the elements in the HTML. But as mentioned earlier, do not include any abundant information that is not needed to identify the target element. Obeying the rules above, try to be consistent with previous step subgoals. Answer with reasoning, but strictly separate the reasoning from the inferred subgoal in the answer. It should be detailed logical reasoning based on the final goal, previous step subgoals, HTML with the elements in it, and importantly the rules above, but without reliance on the current ground-truth action. (Do not include any additional explanation (no matter it is with or without parentheses) as part of the subgoal, even short ones. The subgoal must be just the subgoal with nothing else.) Some random examples of possible subgoals: selecting store location, opening seat options, entering destination, selecting year, entering zip

In-Context Example

HTML

<html> <body> <div main> <div region refine your search>   <button id=0 job category> Job Category </button> <div combobox>   <button id=1 city> City </button> <div combobox>   <button id=2 job type> Job Type </button> <div combobox>   <button id=3 location> Location </button>   <div region facet results block> <select id=4 sort by> <option option most relevant true> Most relevant </option> <option option most recent> Most recent </option> </select>   width= 1 height= 1 /> </body> </html>

Task Search the most recent full time Accounting & Finance Job available in Richmond, Virginia.

Previous intents

opening menu
initiating job search
entering job title
selecting job title
opening state filter
specifying state

Action

Target element: <button id=1 city> City </button>
Action: CLICK

Reasoning

The previous step completed specifying the state. As there is the element for filtering by city now, we can narrow it down by specifying the city, which is required by the final goal ("Richmond"). Also, (a) the inferred subgoal is labeled consistently with the previous steps (especially "opening state filter") and (b) it identifies exactly one target element from HTML.

Intent

opening city filter

In-Context Example ...

In-Context Example ...

Figure 4: The prompt for our intent extractor. We show only one in-context example due to the space limit.

Prompt for LLM Policy

System

You are a helpful assistant that performs actions for solving tasks and completing goals in web environments. Your job is to decide the correct action for the immediate next step. Specifically, you are given the (1) current web page (HTML), (2) final goal, (3) previous subgoals with corresponding actions, (4) suggested subgoals for next step, and (5) target element choices. The (2) final goal is what you ultimately achieve, whereas (3) previous subgoals with corresponding actions is what you have achieved so far. The (4) is a list of suggested subgoals for the immediate next step. The list is sorted from the most probable to the least probable subgoal candidates, but it might contain wrong or irrelevant subgoals as well. You should connect the above inputs together to predict the correct action.

To do that, you first need to choose the correct subgoal from (4). Examine each suggested subgoal from (4), from the beginning of the list (4), and once you find the first subgoal that is valid in the current situation, use it as the subgoal for action prediction. When choosing the subgoal from (4), consider the following aspects:

(a) whether that subgoal makes sense and is irrelevant or already achieved, considering (1), (2), and (3)

(b) what action will be taken given that subgoal and whether it makes sense

Based on the chosen subgoal, predict the correct action to achieve that subgoal. If none of the suggested subgoals make sense or are valid in the current situation, try your best to do action prediction without subgoals.

To make the action prediction, among the (5) given choices of target elements (extracted from the (1) current web page), you need to infer the correct choice and also the action to perform with that target element. Possible actions include {CLICK, SELECT, TYPE}, where SELECT and TYPE require additional "Value" in addition to the action type. With CLICK action, when there are multiple, equally correct choices with the same meaning or intention, you should prioritize elements designed for clicking, such as `<a>`, `<button>`. SELECT action can only be used with `<select>` elements, and its "Value" should be the text that appears between one of the `<option>`-`</option>` pairs. TYPE action is usually performed with `<input>` elements, and its "Value" should be the text to type.

In-Context Example

HTML

<html> <main main> <button button> <svg id=0 img> <title> header.burgerMenu.title </title> <desc> header.burgerMenu.description </desc> </svg> </button> <form> <label> From <input id=1 combobox text departure station, edinburgh (waverley) selected. edinburgh (waverley) enter origin station... /> </label> <fieldset> Out <input id=2 text date use format: 02-apr-23 02-apr-23 /> <select id=3 listbox hour hours> <option 00> 00 </option> <option 01> 01 </option> <option 02> 02 </option> <option 03> 03 </option> <option 04> 04 </option> <option 05> 05 </option> <option 06> 06 </option> <option 07> 07 </option> <option 08> 08 </option> <option 09> 09 </option> <option 10 true> 10 </option> <option 11> 11 </option> <option 12> 12 </option> <option 13> 13 </option> <option 14> 14 </option> <option 15> 15 </option> <option 16> 16 </option> <option 17> 17 </option> <option 18> 18 </option> <option 19> 19 </option> <option 20> 20 </option> <option 21> 21 </option> <option 22> 22 </option> <option 23> 23 </option> </select> </fieldset> <fieldset> <button id=4 button> Same day </button> <button id=5 button> Next day </button> <select id=6 listbox hour hours> <option 00> 00 </option> <option 01> 01 </option> <option 02> 02 </option> <option 03> 03 </option> <option 04> 04 </option> <option 05> 05 </option> <option 06> 06 </option> <option 07> 07 </option> <option 08> 08 </option> <option 09 true> 09 </option> <option 10> 10 </option> <option 11> 11 </option> <option 12> 12 </option> <option 13> 13 </option> <option 14> 14 </option> <option 15> 15 </option> <option 16> 16 </option> <option 17> 17 </option> <option 18> 18 </option> <option 19> 19 </option> <option 20> 20 </option> <option 21> 21 </option> <option 22> 22 </option> <option 23> 23 </option> </select> </fieldset> </form> </main> </html>

Task Book a journey with return option on same day from Edinburg to Manchester on April 2nd and book the best possible option available.

Previous intents and actions

entering departure: [combobox] Departure station, London selected. -> TYPE: edinburg
selecting departure station: [span] Edinburgh (Waverley) -> CLICK
entering destination: [combobox] Arrival station, Sheffield selected. -> TYPE: manchester
selecting destination station: [span] Manchester -> CLICK
selecting journey type: [radio] Return -> CLICK
selecting departure date: [textbox] Date use format: 29-Mar-23 -> CLICK
selecting return date: [link] 2 -> CLICK
entering return date: [textbox] Date use format: -> CLICK
selecting return date: [link] 2 -> CLICK

Suggested intents for action

1. specifying return time
2. confirming return time
3. selecting return time
4. specifying return hour
5. selecting same day

Element choices

A. <svg id=0 img> <title> header.burgerMenu.title </title> <desc> header.burgerMenu.description </desc> </svg>
B. <input id=1 combobox text departure station, edinburgh (waverley) selected. edinburgh (waverley) enter origin station... />
C. <input id=2 text date use format: 02-apr-23 02-apr-23 />
D. <select id=3 listbox hour hours> <option 00> 00 </option> <option 01> 01 </option> <option 02> 02 </option> <option 03> 03
E. <button id=4 button> Same day </button>
F. <button id=5 button> Next day </button>
G. <select id=6 listbox hour hours> <option 00> 00 </option> <option 01> 01 </option> <option 02> 02 </option> <option 03> 03

Action Choice: G Action: SELECT Value: 17

In-Context Example ...

Figure 5: The prompt for our LLM policy with predicted intents. We show only one in-context example due to the space limit.

A.3 Intent Extractor

For discovering intents given demonstration trajectories, we use our intent extractor $\mathcal{M}_{\texttt{extract}}$ powered by GPT-4 (`gpt-4-0125-preview`) (Achiam et al., 2023) for our GPT agents and by Llama-3.1-405B-FP8 (Dubey et al., 2024) for our Llama agents, with the prompt in Figure 4. While we use three in-context examples, we only present one example due to the limited space. The input for actual samples follows the same format as the in-context examples, where the previously discovered intents are used as part of the input for discovery in subsequent time steps.

A.4 Intent Predictor

For training our intent predictor $\mathcal{D}_{\texttt{intent}}$, we augment each of the transitions from the train set of Mind2Web (Deng et al., 2024) with intents discovered with the intent extractor $\mathcal{M}_{\texttt{extract}}$, where the target intent is randomly selected from 5 intents obtained with a temperature of 0.2. Similarly to the dataset augmentation practice by Deng et al. (2024), for each transition from the original trajectory, we form 32 samples with different candidates from the top-80-scoring elements, where 5% of the original train set is excluded for a validation purpose. We employ `Mistral-7B-v0.1` ($\sim$ 7B parameters, license: Apache 2.0, allows research purposes) (Jiang et al., 2023) for fine-tuning with Low-Rank Adaptation (LoRA) (Hu et al., 2021) and `Flan-T5-XL` ($\sim$ 3B parameters, license: Apache 2.0, allows research purposes) for full fine-tuning, on the intent-augmented train set. We estimate approximately 1k GPU hours (Nvidia A100 40GB) are used for training `Mistral-7B-v0.1`, including the exploration and hyperparameter search. For the additional `Flan-T5-XL` training and hyperparameter search, we roughly used 0.4k GPU hours (Nvidia A100 40GB). See Table 6 and Table 7 for the hyperparameter search for `Mistral-7B-v0.1` and `Flan-T5-XL` respectively with best-found values (bold-faced). For intent prediction during the inference phase, we generate up to 5 tokens and use up to 12 beams for $N = 20$ and up to 8 beams for $N = 40$, where the full beam search for each input takes around 1 second.

A.5 LLM Policy

Given the top-k predicted intents, we use a prompt-based LLM policy π for action prediction. We present our prompt for the LLM policy, powered by GPT-4 (`gpt-4-0125-preview`), GPT-3.5 (`gpt-3.5-turbo-0125`), Llama-3.1-70B, and Llama-3.1-405B-FP8 in Figure 5. We incorporate two in-context examples, but due to the limited space, we show only one example. The actual sample input follows the same format as the in-context examples, but we use the in-context examples from a simpler setting (with $N = 7$ element choices) than the actual problem setting (with $N = 20$ or $N = 40$ element choices) to avoid having overly long input contexts. Note that using a smaller N deteriorates the correct element recall and the upper-bound performance as well. We use $k = 5$ top intent predictions for $N = 20$ element choices and $k = 7$ top intent predictions for $N = 40$ element choices.

A.6 Online Evaluation

For the online evaluation, we use a subset of tasks from the Shopping split of the WebArena benchmark (Zhou et al., 2023) with automatic evaluators based on URLs. We leverage our fine-tuned Mistral-7B intent predictors from the Mind2Web experiments without any modifications. As Mind2Web (Deng et al., 2024) does not include stop actions in its dataset, we perform step-wise evaluations to check task completion for all the compared methods. We employ the observation processing and element-ranking model described in Appendix A.2 for all the methods compared in this online evaluation.