Fine-Tuning Web Agents: It Works, But It's Trickier Than You Think

Massimo Caccia[1] **Megh Thakkar**[1,2] **Léo Boisvert**[1,2,3]
Thibault Le Sellier De Chezelles[1,3] **Alexandre Piché**[1] **Alexandre Drouin**[1,2]
Nicolas Chapados[1,2] **Maxime Gasse**[1,2,3] **Alexandre Lacoste**[1]
[1]ServiceNow Research [2]Mila – Quebec AI Institute [3] Polytechnique Montréal

Abstract

Recent advancements in large language models (LLMs) have sparked interest in developing autonomous web agents capable of performing digital tasks through web interfaces in a human-like manner. However, even the strongest closed-source models often struggle to achieve robust results on several benchmarks, while a notable performance gap exists between them and open-source counterparts. This study investigates the potential of fine-tuning to enhance the performance of a smaller, lower-performing but cost-efficient LLM by leveraging successful traces from stronger LLMs, referred to as experts. We outline a comprehensive pipeline for data collection, filtering, and supervised fine-tuning and explore various behavior cloning parameters. Our experiments provide key insights into the challenges of fine-tuning LLMs into web agents on benchmarks like MiniWoB and WorkArena. Notably, we find that the fine-tuned agents' ability to predict expert trajectories does not consistently lead to improved downstream task performance. This raises issues such as off-policy bias and the loss of reasoning abilities during fine-tuning. We discuss potential solutions to these challenges and make both the codebase and a dataset of 140M tokens open-source for the community to build upon.

1 Introduction

Recent advancements in large language models (LLMs) [OpenAI, 2024, Team, 2024] have sparked significant interest in developing autonomous *web agents* capable of performing a variety of tasks through web interfaces in a human-like manner. These agents hold the potential to automate numerous digital tasks, ranging from simple web navigation and form filling—tasks often explored in toy environments—to complex interactions with enterprise software applications, which are predominantly accessed via web browsers. In enterprise settings, such automated agents can, for example, resolve IT incidents or handle data processing in customer relationship management (CRM) platforms. Furthermore, they can be applied to customer-facing tasks like providing personalised restaurant recommendations or booking flights, thereby enhancing user experiences across various domains.

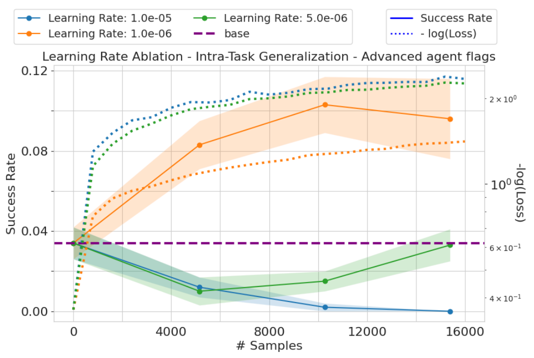


Figure 1: Success rate on WorkArena (left y-axis) and modified likelihood of expert trajectories in the inter-task generalization setup, throughout the fine-tuning phase. The model with the worst ability to predict expert trajectories performs best in this regime. Optimizing the use of expert trajectory data remains a complex and open research challenge.

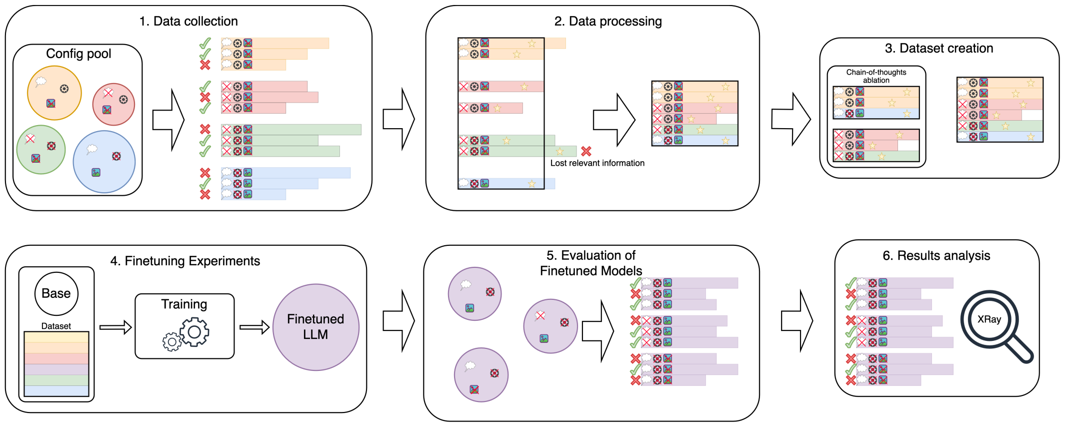


Figure 2: Our generic pipeline: 1) Trajectories are generated using different configurations (Chain-of-thoughts: 💭, use error logs: ⚙, use screenshot: 🖼) and different LLMs (highlighted by different colors). 2) Only the successful trajectories are kept. As each prompt is truncated to fit in our trained model's window, some key information (☆) might get lost in the process. Those samples are discarded. 3) The pipeline now has a pool of data, which can be used to build training sets with different properties. Here, we build an ablation dataset that separates data with and without chain-of-thoughts, and a dataset that merges both. 4) After selecting a dataset, we train our model starting from a base model to make a stronger finetuned LLM. 5) The latter is used along with different agent configurations to assess the finetuning quality. 6) Finally, we can leverage AgentLab's tools to manually analyze the traces produced by the model.

The complexity of tasks explored in the literature varies widely, from basic UI commands on toy web pages, as in benchmarks like MiniWoB [Liu et al., 2018, Shi et al., 2017], to intricate operations on real-world websites [Zhou et al., 2023] and complex enterprise-level interactions [Drouin et al., 2024]. Despite the capabilities of commercial, closed-source LLMs like GPT-4 [OpenAI, 2024] and Claude [Anthropic, 2024], these models often struggle to achieve robust results across various benchmarks [Zhou et al., 2023, Boisvert et al., 2024]. Moreover, their closed-source nature raises concerns about data privacy, security, and cost. Open-source LLMs such as Llama [Meta, 2024] and DeepSeek [DeepSeek-AI, 2024] offer better control and scalability but generally lag behind in performance when acting as web agents [Drouin et al., 2024].

In this study, we investigate the potential of fine-tuning to enhance the performance of smaller, cost-efficient open-source LLMs as web agents. By leveraging successful traces from stronger LLMs (referred to as experts), we aim to bridge the performance gap We outline a comprehensive pipeline for data collection, filtering, and supervised fine-tuning, and explore various behavior cloning parameters.

Our experiments focus on both general web tasks, using benchmarks like MiniWoB [Shi et al., 2017], and more complex, enterprise-related tasks, as in WorkArena [Drouin et al., 2024]. Notably, we find that fine-tuned agents' ability to predict expert trajectories does not consistently lead to improved downstream task performance. This raises issues such as off-policy bias and the loss of pretraining reasoning abilities during fine-tuning. We discuss potential solutions to these challenges and make both the codebase and a dataset of 140 million tokens open-source for the community to build upon.

Our key contributions are:

1. We demonstrate that fine-tuning open-source LLMs to function as web agents can lead to significant performance improvements, although the process is complex with challenges related to task generalization and learning stability.
2. We provide empirical evidence that simply improving the agent's ability to predict expert trajectories does not correlate with better downstream performance, highlighting the need for more sophisticated fine-tuning strategies.
3. We show that the choice of training data, particularly the inclusion of Chain-of-Thought reasoning, plays a crucial role in boosting agent performance.
4. We release a dataset of 140 million tokens of high-quality web agent traces and open-source our codebase to facilitate future research in this area.

2 Web Agent Pipeline

Our experiments rely on an ecosystem of tools for web agents, which we release as open-source contributions to the community to facilitate prototyping, evaluation, training, and reproducibility.

WorkArena[1] (Fig. 5a, § A) is a benchmark for evaluating web agents on the ServiceNow platform [Drouin et al., 2024]. It measures their ability to perform basic tasks using the main UI components of its user interface. For example, one of the tasks consists in filling out a form after receiving the explicit list of desired values for each field. It is the first benchmark to measure the performance of web agents at solving work-related tasks in the enterprise setting.

BrowserGym[2] (Fig. 5b, § A) is a gym environment that facilitates the design and evaluation of web agents in a unified framework. The salient features of BrowserGym include: i) chat-based agent-human interactions, ii) enriched multimodal observations: HTML, AXTree [Zhou et al., 2023], screenshot, set-of-marks [He et al., 2024], element coordinates, etc. and iii) a standard and flexible action space: `click`, `type`, etc..

AgentLab[3] offers a full pipeline for the large-scale evaluation of web agents. It offers features such as parallel evaluation, standardized data collection, and visual trace analysis and inspection tools.

3 Agent Design

We mostly follow the same agent design as Drouin et al. [2024]. For complete details and an example prompt, see § B.

Input Our agents receive the current goal, the current page's accessibility tree[4] (AXTree) [Zhou et al., 2023], and the error message, if any, that resulted from the execution of the previous action. Our study focuses on pure LLM-based agents, hence we do not use screenshot observations.

Prompt We use a tool called dynamic prompting to build our agent's prompt in a modular manner, using different flags to activate or deactivate the desired features. For example, it allows us to activate or deactivate chain-of-thoughts (CoT) reasoning [Wei et al., 2022], which is a technique that encourages language models to generate intermediate reasoning steps to improve their performance on tasks requiring complex problem-solving, rather than directly producing a final answer. It can also activate error logging, or history logging, amongst other things. We use different configurations of those flags in our experiments.

Output Our agent produces a textual reasoning (when CoT is active) plus an action in the form of a function call. We use the high-level action space from BrowserGym, which allows sending messages to the chat, and interacting with the webpage using element identifiers (`bid` attribute).

4 Finetuning Pipeline

The proposed finetuning pipeline for enhancing web agents systematically addresses the challenges of developing models capable of reasoning, planning, and executing complex tasks in enterprise environments. The process is structured into seven key stages:

Step 1: Data Collection We initiate by deploying a collection of web agents within the WorkArena environment. These agents vary across multiple axes, such as the foundational LLMs, observation modalities (AX tree vs. HTML), action spaces (high-level UI actions vs. Python API calls), and prompting techniques like CoT reasoning. Each agent operates in real-world scenarios, collecting interaction traces reflecting diverse approaches to solving tasks like form-filling and list manipulation.

[1] `https://github.com/ServiceNow/WorkArena`

[2] `https://github.com/ServiceNow/BrowserGym`

[3] `https://github.com/ServiceNow/AgentLab`

[4] AXTree is a simplified representation of the page in text format for visually impaired users. It contains about 10x less token than the HTML and it is sufficient for most tasks in WorkArena

The outcome of this step is a comprehensive corpus of agent-generated task traces that encapsulate different strategies for tackling UI interactions.

Step 2: Data Processing The collected data undergoes rigorous filtering and transformation. Successful traces — where agents complete tasks per WorkArena's validation criteria — are retained for training. To adapt the data to finetuning, we simulate interactions using smaller context windows, trimming prompts when necessary. If key information (such as field identifiers) is missing, the trace is discarded. This ensures training data consistency and high relevance to the finetuning models.

Step 3: Dataset Creation Next, we curate multiple datasets for finetuning. One comprehensive dataset includes all traces, while additional ablation datasets focus on specific variables like CoT reasoning. These ablation studies maintain uniform dataset sizes to avoid biasing results due to data quantity, allowing us to isolate the impact of certain features on learning performance.

Step 4: Finetuning Experiments Finetuning is performed on selected base models supported by our framework. We explore two primary experiment types. In *Dataset Ablations*, we finetune the same model across various datasets, maintaining consistent hyperparameters. This experiment evaluates which datasets or features are most beneficial for learning. In *Hyperparameter Ablations*, we hold the dataset constant while systematically varying one hyperparameter (e.g., learning rate) to assess its influence on model generalization. The outcome is a series of finetuned models, saved as checkpoints for further evaluation.

Step 5: Evaluation of Finetuned Models The finetuned models are converted into web agents, and configured with various evaluation flags. These agents are tested on unseen task instances and configurations to assess their generalization abilities across two levels. In *Inter-task Generalization*, we evaluate agent performance on unseen configurations (seeds) of tasks previously encountered during training. In *Cross-task Generalization* we test the agents on entirely novel tasks that were not part of the training set. The resulting evaluation data provides insights into the models' robustness and flexibility in handling diverse enterprise workflows.

Step 6: Analysis of Results The evaluation results are analyzed to extract key insights into model performance. We generate visualizations and plots (as discussed in § 5.1) to highlight trends, such as the impact of different datasets or finetuning strategies on agent success rates. Additionally, tools like AgentLab's `AgentXray` facilitate deeper inspection of agent behaviors, allowing us to identify strengths and weaknesses in decision-making processes. These insights guide further model refinements and inform future research directions.

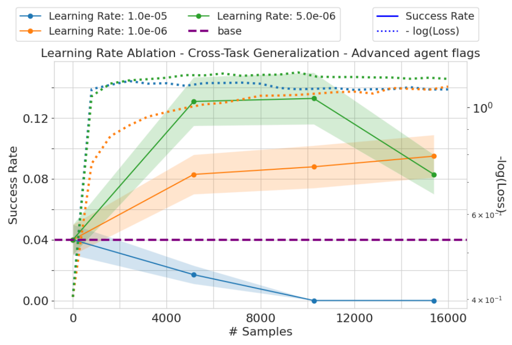
5 Experiments

We evaluate the fine-tuning performance of Llama3.1-8B on the WorkArena and MiniWob benchmarks. Experiments focus on learning rate ablation, dataset ablation (with and without CoT), and generalization to unseen tasks. We measure success rates on held-out task configurations and unseen tasks, using success rate and likelihood of expert trajectories as evaluation metrics. The data collection models are LLAMA3.1-70b and Mistral-Large-2. Additionally, we will fully open-source the WorkArena training dataset, which consists of 32K successful episodes and 140M tokens.

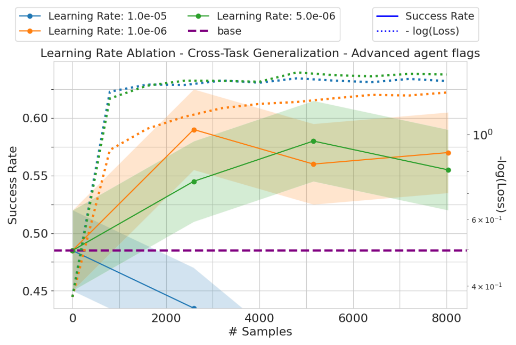
We use two agent configurations: `Advanced` and `Basic`. The `Advanced` configuration employs a more powerful set of prompt options, including CoT reasoning and the ability to generate multiple actions simultaneously. This set of flags was optimized for the Llama-3-8B Instruct model through hyperparameter search. For each reported experiment, we performed two fine-tuning runs and averaged the results. The shaded area represents one standard error in each direction.

5.1 Empirical analysis

In Fig. 1 and Fig. 3, we present a learning rate ablation study on both intra-task and cross-task generalization levels, respectively. For the `Advanced` agents, we observe that in the intra-task generalization regime, the learning rate that results in the worst ability to predict expert trajectories actually yields the best-performing agents on WorkArena. Notably, MiniWoB cannot be evaluated in the intra-task regime due to insufficient task configurations. In the cross-task generalization regime, while all learning rates converge to similar likelihoods of expert trajectories, the model that achieves



(a) WorkArena



(b) MiniWob

Figure 3: Success rate on (left y-axis) and modified likelihood of expert trajectories in the cross-task generalization setup, throughout the fine-tuning phase. Interestingly, the model's improved ability to predict expert trajectories does not directly translate to better performance on downstream tasks.

intermediate predictive accuracy on expert traces catastrophically forgets how to perform tasks in both WorkArena and MiniWoB. Furthermore, in § B.2, we demonstrate that simply training for longer periods and using smaller learning rates does not improve performance. For the results of the `Basic` agents, please refer to § B.3.

This result is quite surprising, as one would expect that a model better at approximating expert behavior in a given situation would also perform better on WorkArena tasks. However, our empirical findings demonstrate otherwise. This raises intriguing research questions: *"How does a model forget how to perform WorkArena tasks while learning to imitate an expert?"* and *"What can we modify, either in the data or training process, to prevent this behavior and thereby develop better web agents?"*. We discuss these questions further in the coming error analysis section and § 8, respectively.

In Fig. 4, we present a dataset ablation study, specifically focusing on Chain-of-Thought (CoT) prompting. Throughout training with the `Advanced` flags, we observe that while the three datasets initially perform similarly, the dataset that strictly uses CoT eventually achieves nearly a 15% success rate increase over the base model and outperforms the next best dataset by almost 5%. This highlights how dataset ablation can guide the iterative process of dataset construction. In the next round of data collection, we might consider allocating the entire data collection budget to agents using CoT prompting.

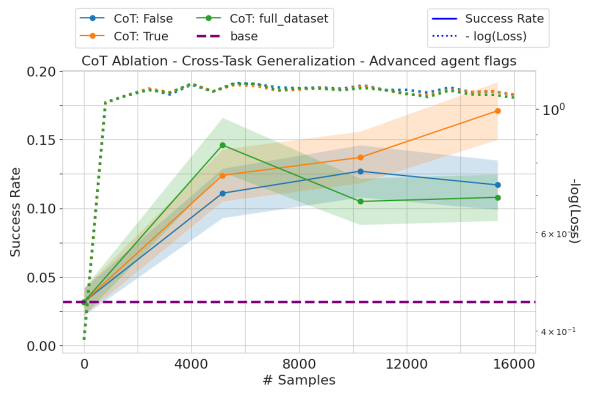


Figure 4: Chain-of-Thought dataset ablation on WorkArena. Training strictly on expert data using CoT produces the best fine-tuned agents.

5.2 Analysing the Finetuned Model's behaviour

When performing fine-tuning and particularly evaluating on unseen tasks, we observe that the model is in-fact able to imitate the behavior based on the traces of the training data. For example, we see that the model struggles significantly with tasks requiring navigation either due to not having encountered them at all during training or having observed very similar navigation tasks, failing to acquire the skill to solve them. Parallely, the model becomes powerful at previously unseen form filling tasks, having observed and learnt from similar tasks in the training data. These observations indicate that the model is able to imitate, or learn with fine-tuning on individual observation-action instances and apply them for sequential decision making. Interestingly, it is often unable to improve over types of tasks that were impossible for the base model to solve.

Error Analysis As mentioned previously, the validation loss alone is insufficient to predict downstream task success. To understand why, we explored the traces from agent $1e-05$ in Fig. 3 at the first 3 checkpoints of its training, respectively at 5,120, 10,240 and 15,360 samples. At 5,120 samples, the model performs best, while the performance deteriorates markedly in subsequent checkpoints.

For instance, in all instances of a task asking the agent to change the current user, the model appears to have memorized examples rather than reasoning through the task. At both 10,240 and 15,360 samples, all the initial actions consistently start with `click(a324)`, which is the action provided as an example in the prompt. This suggests the model is not distinguishing between the observed data and the examples, highlighting a deficiency in its reasoning capabilities. Additionally, for these 2 checkpoints, very few traces include a "think" step.

At 10,240 samples, the model manages to complete only one task—likely by coincidence. This task, the dashboard task, resembles a standard question-answering task and requires minimal agentic abilities, mostly involving reading from the AxTree. A notable observation is that the model no longer generates "think" steps at this stage, jumping directly into actions without reasoning.

By 15,360 samples, the model's reasoning capabilities remain absent. It continues to default to actions from the examples, such as frequently outputting the bid "a324" from the training set, further reinforcing that the model is failing to adapt its actions based on actual observations. Overall, the progression from 5,120 to 15,360 samples indicates a significant decline in the model's agentic and reasoning abilities, with increasing reliance on memorized examples rather than understanding the task context and observations.

6 Related Works

LLM-based Web Agents AI researchers have long pursued the development of autonomous agents, and recent advances in LLMs, combined with improvements in reasoning and planning capabilities, suggest that this goal within reach [Wei et al., 2022, Yao et al., 2023, 2024]. One prominent class of agents focuses on equipping a powerful model with defined action and observation spaces [Zhou et al., 2023, Deng et al., 2023, Drouin et al., 2024]. Alongside using already trained models to mimic agentic behavior, multiple different fine-tuning methods have been suggested to improve the domain-specific performance of using models as agents. In WebLinx [Lù et al., 2024], authors have used supervised fine-tuning based on expert traces to predict the next action to take in multi-step interactions. Recent approaches [Putta et al., 2024, Song et al., 2024] generate successful and unsuccessful example traces to fine-tune their agent with preference optimization methods like Direct Preference Optimization [Rafailov et al., 2024]. Finally, other methods have used reinforcement learning approaches for fine-tuning agents [Zhou et al., Zhai et al., 2024].

Off-Policy Bias & Knowledge Distillation Knowledge Distillation [Bucila et al., 2006, Hinton, 2015] transfers knowledge from a larger "teacher" model to a smaller "student" model. In sequential decision-making tasks like web agents, this process introduces *off-policy bias* when the student is trained on expert trajectories generated by the teacher. The student may struggle during inference when encountering states resulting from its own actions, leading to error accumulation due to the mismatch in state-action distributions [Ross et al., 2010].

To mitigate off-policy bias, methods such as DAgger [Ross et al., 2010] collect data by rolling out the student policy and aggregating it with previous datasets, enabling the model to learn from states it is likely to encounter during inference. An analogous issue in supervised learning is *exposure bias* [Ranzato et al., 2015, Bengio et al., 2015], where models trained with teacher forcing on ground-truth tokens face distribution mismatch during inference when relying on their own outputs. Techniques like *Scheduled Sampling* [Bengio et al., 2015] and methods that minimize divergence between student and teacher models [Gu et al., 2023, Agarwal et al., 2024] help mitigate this mismatch by better simulating inference conditions or training the student on its own generated sequences.

Knowledge Distillation (KD) [Bucila et al., 2006, Hinton, 2015] has emerged as a successful technique for transferring the knowledge of a larger or more complex model to a more efficient model. Distillation involves training a "student" model, which is generally smaller, to replicate the behavior of a "teacher" model, which is generally larger. In text generation, distillation approaches have attempted to either train the student model for token-level predictions using outputs of the teacher model [Sanh et al., 2019], or train the student model to make predictions at the sequence-level [Kim and Rush, 2016, Chiang et al., 2023, Peng et al., 2023]. However, distilling knowledge from a significantly more capable model to a student model often has complications, often leading to exposure bias when there is a difference in training and inference settings for the student model [Ranzato et al., 2015, Bengio et al., 2015].

7 Discussion & Future Work

Our findings suggest that the decline in model performance, despite improved trajectory prediction, is due to two key factors: (1) *off-policy bias* from training on expert traces, and (2) *catastrophic forgetting* of reasoning abilities learned during pretraining. Our future work will focus on addressing these challenges to develop more robust web agents.

Addressing Off-Policy Bias: Training solely on expert trajectories introduces off-policy bias, as it fails to represent the conditions encountered during real-world task execution. To mitigate this, we propose using *Iterative Self-Improvement*, where traces generated by the fine-tuned model are iteratively added to the training data. This allows the model to learn from its own mistakes, improving adaptability and reducing reliance on expert-only traces. For *Dataset Diversification*, we aim to train on a more diverse subset of tasks to prevent premature overfitting. By exposing the model to varied task types and complexities, it can extract more relevant knowledge. Additionally, *model-based learning* will focus on predicting environmental state transitions, reducing overfitting to specific expert actions.

Preventing Catastrophic Forgetting: Fine-tuning can lead to the loss of general reasoning capabilities acquired during pretraining. To counteract this, we propose *Model Merging*, periodically merging the fine-tuned model with the base model by interpolating their parameters. This approach helps retain general reasoning skills while incorporating new task-specific knowledge. Additionally, *Continued Pretraining* on a mix of original and new task-specific data will maintain emergent reasoning capabilities, ensuring the model retains its general problem-solving skills

8 Conclusion

We investigated fine-tuning open-source LLMs to function as web agents on benchmarks like MiniWoB and WorkArena. While fine-tuning led to performance gains, challenges such as task generalization, learning instability, and off-policy bias persist. Notably, improved prediction of expert trajectories did not consistently enhance downstream task performance, suggesting that imitation learning alone may be insufficient.

Our results emphasize the importance of training data composition, particularly the inclusion of Chain-of-Thought reasoning, in boosting agent capabilities. Addressing off-policy bias and preventing catastrophic forgetting are critical for future work. We release our codebase and a dataset of high-quality web agent traces to facilitate further research toward more robust web agents.

References

R. Agarwal, N. Vieillard, Y. Zhou, P. Stanczyk, S. R. Garea, M. Geist, and O. Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024.

Anthropic. The Claude 3 Model Family: Opus, Sonnet, Haiku, 2024. URL `https://www.anthropic.com/news/claude-3-family`.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

L. Boisvert, M. Thakkar, M. Gasse, M. Caccia, D. Chezelles, T. Le Sellier, Q. Cappart, N. Chapados, A. Lacoste, and A. Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. *arXiv preprint arXiv:2407.05291*, 2024.

C. Bucila, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Knowledge Discovery and Data Mining*, 2006. URL `https://api.semanticscholar.org/CorpusID:11253972`.

W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL `https://arxiv.org/abs/2405.04434`.

X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su. Mind2Web: Towards a generalist agent for the web. *arXiv*, abs/2306.06070, 2023.

A. Drouin, M. Gasse, M. Caccia, I. H. Laradji, M. D. Verme, T. Marty, L. Boisvert, M. Thakkar, Q. Cappart, D. Vazquez, N. Chapados, and A. Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks?, 2024.

Y. Gu, L. Dong, F. Wei, and M. Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

H. He, W. Yao, K. Ma, W. Yu, Y. Dai, H. Zhang, Z. Lan, and D. Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. *arXiv*, abs/2401.13919, 2024. URL `https://arxiv.org/abs/2401.13919`.

G. Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mixtral of experts, 2024.

G. Kim, P. Baldi, and S. McAleer. Language models can solve computer tasks. *arXiv*, abs/2303.17491, 2023. URL `https://arxiv.org/abs/2303.17491`.

Y. Kim and A. M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, 2016.

E. Z. Liu, K. Guu, P. Pasupat, T. Shi, and P. Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018.

X. H. Lù, Z. Kasner, and S. Reddy. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*, 2024.

Meta. Llama 3: Meta's latest large language model. `https://github.com/meta-llama/llama3`, 2024. Accessed: 2024-06-03.

OpenAI. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

B. Peng, C. Li, P. He, M. Galley, and J. Gao. Instruction tuning with gpt-4. *ArXiv*, abs/2304.03277, 2023. URL `https://api.semanticscholar.org/CorpusID:257985497`.

P. Putta, E. Mills, N. Garg, S. Motwani, C. Finn, D. Garg, and R. Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.

R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2010. URL `https://api.semanticscholar.org/CorpusID:103456`.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019. URL `https://api.semanticscholar.org/CorpusID:203626972`.

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

Y. Song, D. Yin, X. Yue, J. Huang, S. Li, and B. Y. Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. URL `https://aclanthology.org/2024.acl-long.409`.

L. Team. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf`.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv*, abs/2210.03629, 2023. URL `https://arxiv.org/abs/2210.03629`.

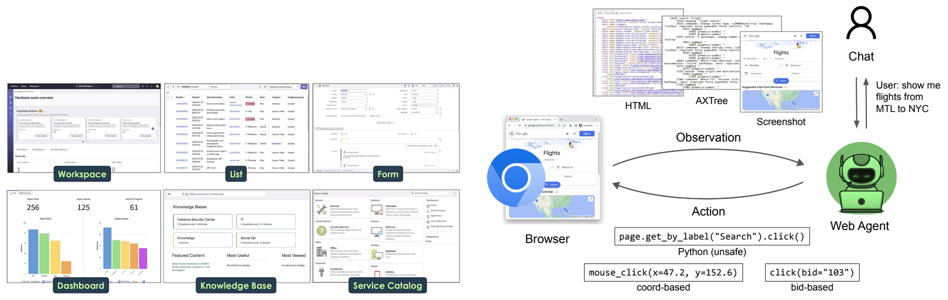
S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Y. Zhai, H. Bai, Z. Lin, J. Pan, S. Tong, Y. Zhou, A. Suhr, S. Xie, Y. LeCun, Y. Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *arXiv preprint arXiv:2405.10292*, 2024.

S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, Y. Bisk, D. Fried, U. Alon, and G. Neubig. Webarena: A realistic web environment for building autonomous agents. *ArXiv*, abs/2307.13854, 2023. URL `https://arxiv.org/abs/2307.13854`.

Y. Zhou, A. Zanette, J. Pan, A. Kumar, and S. Levine. Archer: Training language model agents via hierarchical multi-turn rl. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

A Software



`page.get_by_label("Search").click()`

`mouse_click(x=47.2, y=152.6)`

`click(bid="103")`

(a) WorkArena **(b)** BrowserGym

Figure 5: (a) WorkArena is a collection of tasks which measure the ability of web agents to interact with basic UI components in the ServiceNow platform. (b) BrowserGym is a framework to execute web agents that receive a natural-language goal from a human user via chat, perceive the environment (web browser) through a set of multimodal observations (e.g., HTML and screenshot), and control it via a standardized set of available actions.

B Agent Design

Below are the general design choices of our LLM-based web agent with chain-of-thought prompting [Wei et al., 2022].

Language models: Our study focuses on open-source LLMs. For data collection we use both **Llama3.1-70b** [Meta, 2024] (`meta-llama-3.1-70b-instruct`, 70B parameters, 128K context) and **Mistral Large 2** [Jiang et al., 2024] (`mistral-large-2407`, 123B parameters, 128K context). For fine-tuning we consider a smaller **Llama3.1-8b** [Meta, 2024] model (`meta-llama-3.1-8b-instruct`, 70B parameters, 128K context). These LLMs are deployed using Hugging Face's Text Generation Inference (TGI) library on 4 A100 GPUs.

Observation space: Our observation space is composed of the goal, the current page's HTML and/or AXTree,[5] the currently focused element, and the error from the previous action if any. We also augment each element with two extra boolean properties provided by BrowserGym, `clickable` and `visible`.

Action space: We use BrowserGym's high-level action space with `chat` and `bid` primitives [Drouin et al., 2024] which respectively allow the agent to send messages to the chat (‘`send_msg_to_user(text)`‘, necessary for information retrieval tasks), and to interact with the page's HTML elements using their unique identifiers (e.g., `click(bid)`, `type(bid, text)` etc.). The `bid` primitives rely on the unique `bid` attribute given by BrowserGym to each HTML element, which is made available textually in the HTML and AXTree. The full action space is described to the agent in the prompt, with individual examples of valid function calls for each primitive. For an example prompt, see Fig. 6.

History: To extend the horizon window of our agent, at each time step we re-inject into the agent's prompt the history of all previous actions and thoughts (from chain-of-thought) since the start of the episode. This gives our agent a chance to recall its previous thoughts, thereby providing a crude memorization mechanism to otherwise memory-less agents.

Zero-shot examples: In the prompt, we provide a single generic example of how the chain-of-thought and action outputs should be formatted. This contrasts with other methods [Kim et al., 2023] where task-specific few-shot examples are provided, yet aligns with our objective of developing zero-shot agents able to solve a large range of new tasks.

[5]On WebArena and WorkArena we only use AXTrees because HTML is prohibitively large. On MiniWoB we use both AXTree and HTML as it consistently gives the best performance.

Example Prompt - Order Sales Laptop task

```
# Instructions
Review the current state of the page and all other information to find the best
possible next action to accomplish your goal. Your answer will be interpreted
and executed by a program, make sure to follow the formatting instructions.

## Goal:
Go to the hardware store and order 6 "Sales Laptop" with configuration
{'Additional software requirements': 'Slack, Zoom, Google Workspace, HubSpot, Adobe Creative Cloud',
'Adobe Acrobat': True, 'Adobe Photoshop': False, 'Microsoft Powerpoint': False, 'Siebel Client': False}

# Observation of current step:

## AXTree:
Note: [bid] is the unique alpha-numeric identifier at the beginning of lines for each element in the
AXTree. Always use bid to refer to elements in your actions.
Note: You can only interact with visible elements. If the "visible" tag is not
present, the element is not visible on the page.

RootWebArea 'Catalog | ServiceNow'
...
    [a] Iframe 'Main Content', visible
      RootWebArea 'Catalog', focused
...
                      [a251] heading 'Hardware', clickable, visible
                        [a252] link 'Hardware', clickable, visible
...
                [a261] link '', clickable, visible
                  [a262] table '', visible
                    [a263] rowgroup '', visible
                      [a264] row '', visible
                        [a265] gridcell '', visible
                        [a268] gridcell 'Hardware. Order from a variety of hardware to meet your business
                        needs, including phones, tablets and laptops. Order from a variety of hardware to meet
                        your business needs, including phones, tablets and laptops.', clickable, visible
                          [a269] link 'Hardware. Order from a variety of hardware to meet your business
                          needs, including phones, tablets and laptops.', clickable, visible
                            [a270] heading 'Hardware', visible
...

## Focused element:
bid='a85'

# History of interaction with the task:
...

# Action space:

Note: This action set allows you to interact with your environment. Most of them
are python functions executing playwright code. The primary way of referring to
elements in the page is through bid which are specified in your observations.
13 different types of actions are available.
...
fill(bid: str, value: str)
    Description: Fill out a form field. It focuses the element and triggers an input event with the
    entered text. It works for <input>, <textarea> and [contenteditable] elements.
    Examples:
        fill('237', 'example value')
        fill('45', 'multi-line\nexample')
        fill('a12', 'example with "quotes"')
...
send_msg_to_user(text: str)
    Description: Sends a message to the user.
    Examples:
        send_msg_to_user('Based on the results of my search, the city was built in 1751.')

Only a single action can be provided at once. Example:
fill('a12', 'example with "quotes"')
...
# Concrete Example

Here is a concrete example of how to format your answer.
Make sure to follow the template with proper tags:

<think>
From previous action I tried to set the value of year to "2022",
using select_option, but it doesn't appear to be in the form. It may be a
dynamic dropdown, I will try using click with the bid "a324" and look at the
response from the page.
</think>

<action>
click('a324')
</action>
```

Figure 6: Example prompt of our LLM-based agent. Some parts are truncated (...) for clarity.

Parse and retry: Once the LLM provides an answer, we have a parsing loop that can re-prompt the agent up to 4 times to make it aware of a parsing mistake. This can save the agent from making basic mistakes and is mainly useful for less capable LLMs. Once parsed, the action is executed via BrowserGym, which moves to the next step.

Prompt truncation: When the prompt is too large for the context window of our agent, we progressively truncate the HTML and AXTree from the end until it fits the maximum allowed number of tokens.

B.1 More results

B.2 Basic Agents' results

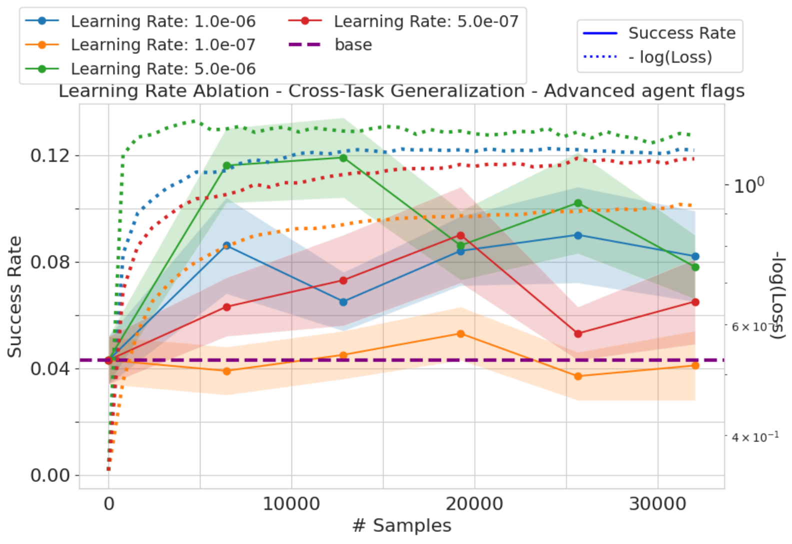


Figure 7: Learning rate ablation on WorkArena in the cross-task generalization setup. Training longer or using smaller learning rates does not improve downstream performance, indicating that these strategies may not enhance the agent's generalization to unseen tasks.

B.3 Basic Agents' results

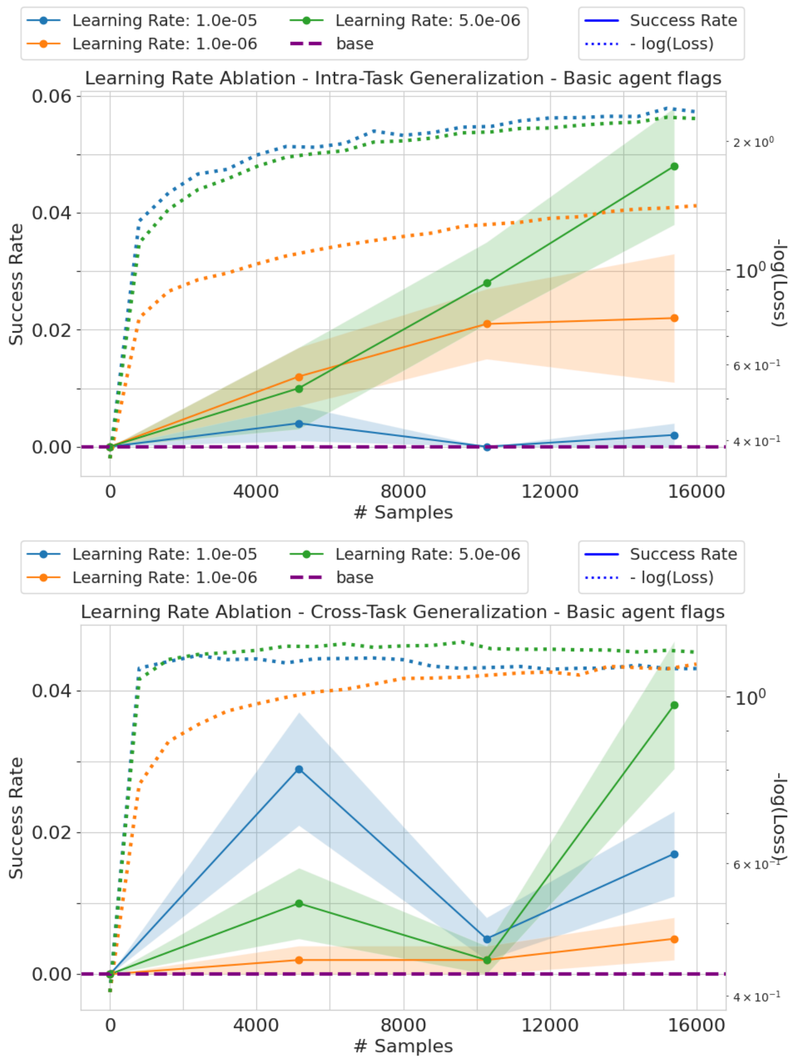


Figure 8: Learning rate ablation for the Basic agent in WorkArena