Speculations on Training General Web Agents via Neural Environment Generation

James Chen

March 4, 2025

Introduction

This document proposes a research direction for training general web agents using realistic neural website simulations. The key insight is leveraging grounding from real websites, employing large language models (LLMs), and reinforcement learning (RL) techniques.

Core Arguments

The main arguments discussed are:

(1) Neural generation of realistic websites and simulated browsing behaviors is feasible.

(2) Such generation does not require fully accurate trajectories.

(3) A resulting neural world model can potentially train "good enough" general web agents.

1 Generating Realistic Neural Websites

Over the past year, we've developed engineering tools with the following properties:

- Randomly walk a website, grouping similar actions based on HTML-derived features into deterministic action "sets."
- Use these sets to classify pages into equivalence classes (ECs), grouping pages by function rather than visual similarity alone.
- Deterministically relate actions to transitions between ECs.

Although initial attempts at naive textual generation from LLMs resulted in unrealistic pages, our method grounds LLM generation through ECs and deterministic action-transition dynamics derived from real websites. This approach ensures diversity and realism in synthetic environments.

Technical Clarification: For example, our similarity measure groups product pages on Amazon based on common actions ("add to cart," "buy now," "one-time purchase"). Pages lacking these actions (like settings pages) are grouped separately. During generation, known actions from real websites ensure realistic state transitions and content.

Realistic HTML generation is also achievable, either through caching pages from ECs or enforcing known action frequency thresholds. The simulation can deterministically manage URLs and site-specific contexts.

2 Trajectory Accuracy and Data Collection

High accuracy of individual simulated trajectories isn't strictly necessary. By observing user interactions (clickstreams), our tools capture sufficient information to create diverse and functional environments. Although deep or delayed action effects (e.g., toggles affecting EC2 configurations) are initially opaque, longer-term observations gradually reveal and label these effects. This process requires only retrospective labeling after observing full trajectories, without needing goal-specific interactions. As long as the generated synthetic data is consistent and diverse, we can specialize on real websites down the line.

3 Training Web Agents

Agents trained in these neural environments can employ methods similar to Deepseek-R1, leveraging rejection sampling to filter unrealistic transitions. An important advantage is maintaining consistent internal representations mirroring realistic website states (e.g., items in an Amazon cart), simplifying rejection sampling and supporting arbitrary initial states.

4 Considerations and Challenges

Note: I almost implicitly assume a PRM here to prune out drastically terrible 'succesful' trajectories, as well as failures.

Tangential Considerations

The engineering tools described initially targeted site-specific tasks but revealed limited commercial interest without significant UX innovation. However, HTML-based similarity functions, tuned per website, have empirically served as effective proxies for page functionalities, allowing automated grouping into meaningful equivalence classes. These classes facilitate enforcing sufficient realism and diversity in synthetic environments without perfect accuracy.

Although deterministic server-based web environments with fixed rewards seem unscalable, neural environments could naturally support few-shot, test-time training and potentially improve open-ended agent capabilities.

Broader Insights and Analogies

Empirical evidence suggests that effective web agents require partial observability handling, particularly recovering from unobserved errors. Current LLM baselines lack the sharp biases necessary for effective RL-driven learning from live web environments. This behavior needs to be somehow honed for robustness.

A neural approach enables controlled curriculum learning, progressively sharpening error-recovery behaviors essential for robust web agents. Diverse, increasingly complex environments prevent models from overfitting and remembering certain non-generalizable strategies, emphasizing necessary error-recovery skills.

It's rather unclear if the current approach of SFT collection will be enough in the short-to-mid term to create enough of this bias to start RLing on real websites.

Viewing this approach as a multi-agent scenario—one neural world-model agent and a traditional LLM agent—offers an intriguing direction for fostering emergent behaviors.

Curriculum Holes for Higher Entropy Problems

A slight concern I have (though perhaps unfounded) is the existence of reasoning 'curriculum holes' for higher entropy problems like open-ended reasoning. A way to think about this is

- High entropy: Reasoning over web pages has a lot of possible explanations for some arbitrary intent, and trajectories an agent has to be 'forced down' to correct errors

- Low entropy: Reasoning over math problems has fewer possible explanations for the right answer, and you can simply ignore incorrect reasoning/action sections in context without (the agent doesn't have to 'restore' the environment itself.)

I think it could be that math and code has a natural complexity buildup in a narrow direction, while this is less true for higher entropy problems like open-ended computer use which may potentially have holes in complexity in very broad directions. This may limit robustness.

This is purely speculative, but human hunter-gatherer instinct of backtracking from bad situations is poorly reflected in datasets, and difficult to just hard-scale by buying data from data farms. It's probably harder to find than say, humor.

We're lucky that it's pretty expressible in language (in comparison to say like, reasoning over the physical world's dynamics) as computer screens are still pretty discrete. An assumption you'd be taking with this approach is some cold start data + neural environment scaling is a more viable direction than whatever the big labs are currently doing, and let's you sharpen this backtracking behavior.

In all...

1. It's very feasible to generate realistic websites by grouping HTML actions into equivalence classes and chaining them deterministically.
2. Neural environments simplify rejection sampling, support realistic online trajectories, and allow consistency across multiple interaction steps.
3. Individual trajectory accuracy isn't critical; diversity, realism, and gradual observation-based labeling suffice.
4. This method enables controlled curriculum learning, progressively honing partial observability and error-recovery behaviors essential for robust agents.
5. There's a speculative concern regarding potential "curriculum holes" in complexity, especially for open-ended, high-entropy reasoning tasks.
6. Overall, I've provided reasoning and analogies justifying why neural website generation could work, why it's promising compared to current methods, and why it's potentially necessary for robust web agent training.