GTA: Generating Long-Horizon Tasks for Web Agents at Scale

Tenghao Huang[1,2†], **Kung-Hsiang Huang**[2], **Prafulla Kumar Choubey**[2]
Yilun Zhou[2], **Muhao Chen**[3], **Jonathan May**[1], **Chien-Sheng Wu**[2]
[1]University of Southern California, [2]Salesforce AI Research
[3]University of California, Davis
† Work done during internship at Salesforce.
tenghaoh@usc.edu

Abstract

Web agents, which couple language models with browsing and tool-use capabilities, show promise as open web assistants. Yet progress is increasingly limited by the lack of scalable, process-level supervision. Existing benchmarks are largely *manually constructed*, providing only coarse start–goal annotations without intermediate trajectories, while recent automatic generation efforts remain expensive, biased, and shallow. These limitations prevent reliable training and evaluation of agents that must generalize to realistic, multi-hop, cross-page tasks. We introduce a scalable framework **GTA** that integrates crawling, retrieval-based seeding, in-context generation, and automated quality control to produce realistic tasks paired with executable trajectories. This design decouples crawling from generation for greater efficiency, grounds tasks in the site graph to enforce compositionality, and ensures dense supervision through deterministic replays and systematic validation. We instantiate the pipeline on over 50 websites covering e-commerce, government, forums, and news, with multilingual and multi-hop coverage. The resulting benchmark reveals a significant human–agent performance gap and enables detailed diagnostics. Our contributions are threefold: (i) formalizing multi-hop web-agent task generation, (ii) proposing an efficient and validated pipeline for automatic data creation, and (iii) releasing a self-evolving benchmark ecosystem where end users can generate up-to-date tasks grounded in live web content [1].

1 Introduction

Web agents—language models coupled with a browser and tool-use interfaces—have rapidly emerged as a core direction for building general-purpose assistants that can search, navigate, and act on the open web (Shi et al., 2017; Liu et al., 2018; Zhou et al., 2024; Huang et al., 2025a,c). Despite striking demonstrations, progress is increasingly constrained by data. Existing benchmarks provide only human-annotated task descriptions paired with a curated end-to-end solution, offering no supervision over the agent's latent decision process (Deng et al., 2023; Wang et al., 2023; Zhou et al., 2024; Spangher et al., 2025; Huang et al., 2025b). This is problematic: prior work has highlighted that web navigation is naturally a hidden Markov process, where the critical uncertainty lies in the unobserved intermediate states and actions (Sodhi et al., 2024; Huang et al., 2025a). Without reliable ground-truth trajectories, agents are overfitting to small hand-authored sets, and struggling to generalize to realistic multi-hop, cross-page use cases (Murty et al., 2025; Ishmam and Marino, 2026).

To address the scarcity of large-scale data, recent efforts have begun to generate web tasks automatically. For instance, AgentTrek (Xu et al., 2024) converts online tutorial documents into executable trajectories and validates them with a vision-language agent, while WebWalker (Wu et al., 2025) and NNetNav (Murty et al., 2025) use an LLM-based policy model to explore websites and then label the resulting trajectories with intent. While promising, these approaches face three critical limitations. First, they are **prohibitively expensive in both time and cost**: constructing a single task requires repeated agent–environment rollouts and multiple LLM calls, driving costs upward and limiting throughput. Second, they **induce strong exploration bias**: LLM policies tend to overfit to "obvious" paths (e.g., repeatedly selecting product detail pages), while overlooking more nuanced or less salient site functionalities such as rewards programs, sales, or new arrivals. As a result, large portions of the site graph remain unexplored and untested. Third, they **lack compositionality and utility**: most tasks gener-

[1]Code available at https://github.com/tenghaohuang/GTA.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 18805–18820

ated by existing approaches collapse to single-hop retrieval or form-filling that could be solved by a simple search query, failing to capture the multi-hop, cross-page dependencies of real-world navigation. As a result, such tasks offer little practical utility for users, since solving them does not require deeper reasoning, sustained interaction, or problem-solving skills that would actually test or benefit an intelligent web agent in realistic settings. We provide detailed comparison in Tab. 1.

We introduce **GTA** (**G**enerate Long-Horizon **T**asks for Web **A**gents at Scale), a scalable task-generation and benchmarking framework for web agents that integrates *crawl*, *retrieve*, *in-context generation*, and *quality control* to produce realistic, graded-difficulty tasks paired with *executable ground-truth trajectories*. Concretely, we first crawl publicly accessible sites to construct a content/link graph; next, we retrieve candidate pages and evidence snippets as seeds; we then generate tasks via in-context prompting that references on-site facts, UI elements, and workflows; we filter with an LLM-based quality control protocol that enforces clarity and solvability against the current crawl; and finally we record a minimal, programmatically executable navigation path (clicks, scrolls, inputs) enabling deterministic replays and step-level attribution.

This design directly targets the three challenges identified earlier. *Efficiency and cost*: we separate one-time crawling from lightweight, retrieval-seeded task generation, eliminating repeated policy rollouts and reducing both redundant LLM calls and wall-clock latency. *Exploration coverage*: by leveraging retrieval-based seeding rather than policy-driven exploration, our method avoids the strong biases of LLM navigators and systematically samples diverse, semantically rich regions of each website's graph. *Compositionality and utility*: grounding generation in the site graph enables the construction of multi-hop tasks that require reasoning across interconnected pages, mirroring the dependencies and open-ended objectives characteristic of real user workflows.

To automatically generate diverse multi-hop web agent tasks, we distinguish between two primary settings: **intra-website queries**, where reasoning unfolds entirely within a single domain, and **inter-website queries**, which require coordination across multiple domains. Within intra-website tasks, our benchmark spans a broad range of languages, including English, Italian, Japanese, German, and Chinese, ensuring coverage beyond English-centric environments.

We instantiate the pipeline on over **50** public websites spanning e-commerce catalogs, entertainment/government pages, forums, and finances, etc. Beyond automatic filtering, we collect human annotations to assess answer correctness, complexity, utility and realism, confirming that tasks are reasonable for humans yet challenging for current agents. Our contributions are three-fold:

- **Problem formulation.** We introduce and formally define the new task of *multi-hop web-agent task generation*, where tasks require agents to reason across multiple pages or domains rather than collapsing to trivial information lookup.

- **Efficient automatic data generation.** We propose an efficient pipeline that markedly reduces per-task time and cost while producing challenging tasks; each task includes an executable minimal "gold path" for deterministic replay and step-level diagnostics, safeguarded via verifier checks and negative controls.

- **Evolving benchmark.** Rather than treating **GTA** as a fixed dataset, we frame it as a self-evolving benchmark ecosystem in which end users can continually generate tasks from current web content. Coupled with our verification pipeline, **GTA** preserves task quality and reproducibility while exposing agents to a changing, realistic task distribution, thereby reducing overfitting and enabling ongoing evaluation of generalization.

2 Benchmark Design

The goal of our benchmark is to create web-agent tasks that systematically capture the *multi-hop* nature of realistic online interactions. Unlike prior datasets that treat each task as a linearized trajectory, we explicitly ground task generation in the structural properties of the underlying web graph. This design choice serves two objectives: (i) it reflects the fact that solving many real-world queries requires reasoning across multiple, interlinked webpages, and (ii) it enables reproducible task construction decoupled from transient agent rollouts. In what follows, we describe the full pipeline in four stages: collecting observations by

Dataset	Multi-hop	Scalable Auto-gen	Groundtruth	Dynamic	Multilingual
WebVoyager (Wang et al., 2023)	✗	✗	✓	✗	✗
WebArena (Zhou et al., 2024)	✗	✗	✓	✗	✗
WebWalker (Wu et al., 2025)	✓	✗	✓	✗	✓
BrowseComp (Wei et al., 2025)	✓	✗	✓	✗	✗
WebDS (Hsu et al., 2025)	✓	✗	✓	✗	✗
AgentTrek (Xu et al., 2024)	✗	✓	✗	✗	✗
NNetNav (Murty et al., 2025)	✓	✓	✗	✗	✗
GTA (Ours)	✓	✓	✓	✓	✓

Table 1: Comparison of benchmark properties. **GTA** is the only benchmark that combines *automatic task generation*, *multi-hop reasoning*, *executable ground-truth*, *dynamic expansion*, and *multilingual coverage*.

crawling and constructing a site graph, generating multi-hop tasks through retrieval-augmented prompting, controlling task quality via systematic verification, and analyzing coverage relative to existing benchmarks.

2.1 Benchmark Construction

Defining Multi-hop Tasks. We aim to construct tasks that require *multi-hop reasoning* over web environments. Formally, let $\mathcal{G} = (V, E)$ denote a directed web graph, where each node $v \in V$ corresponds to a crawled webpage and each edge $(v_i, v_j) \in E$ corresponds to a hyperlink from v_i to v_j. A task is represented as $T = \langle q, A, P^\star \rangle$, where q is the natural-language query, A is the answer, and $P^\star = \langle v_1, v_2, \ldots, v_m \rangle$ is a minimal executable path within $\mathcal{G}$ that leads to A.

We define a task as *multi-hop of order* n if solving it requires aggregating information from at least n distinct webpages, i.e.,

$$|\{v \in P^\star : \text{evidence}(v) \neq \emptyset\}| \geq n, \quad n > 1.$$

Here, evidence(v) denotes the subset of page content from node v that is necessary to derive the correct answer. In contrast, a single-hop task admits $n = 1$, solvable by local reasoning from one node only.

Collecting Observations. To instantiate $\mathcal{G}$, we perform breadth-first crawling over publicly accessible domains. Each hyperlink discovered from page v is initially added as a child node. However, paths to a given webpage are not unique, which risks cycles and inflated traversal lengths. To address this, we enforce a canonical shortest-path policy: whenever a node is reached via a shorter path, we update its parent assignment accordingly. Specifically, each node stores the accessibility tree of its corresponding webpage.

Generating Multi-hop Tasks. Once $\mathcal{G}$ is collected, we embed each node $v \in V$ into a dense representation $h_v \in \mathbb{R}^d$ using an LLM-based encoder f_θ. Given a seed node v_s, we retrieve a candidate set $\mathcal{R}(v_s) = \{v_1, \ldots, v_k\}$ via dense retrieval. To induce compositionality, we randomly select a secondary node $v_t \in \mathcal{R}(v_s)$ and provide $(\text{content}(v_s), \text{content}(v_t))$ as context to a generator LLM. The model is prompted to synthesize a query q whose answer A requires reasoning across both pages. This retrieval-and-pairing strategy enforces multi-hop dependencies by design, rather than relying on incidental cross-references.

Controlling Quality. All generated tasks are subjected to a multi-stage verification pipeline:

1. *Multi-hop validation.* We reject cases where the model produces a conjunction of two independent single-hop queries (e.g., linked by "and"), which do not constitute genuine compositional reasoning.
2. *Answer correctness.* For each candidate (q, A), we condition a verifier LLM on the content of supporting nodes only, requiring it to reproduce or validate A. Incorrect or unverifiable outputs are discarded.
3. *Ambiguity detection.* We assess whether q admits a unique answer. Queries missing critical qualifiers (e.g., temporal or geographic constraints) are filtered out.
4. *Solvability check.* We ensure the existence of at least one minimal executable path $P^\star$ within $\mathcal{G}$ such that traversal yields sufficient evidence to derive A. This guarantees deterministic replayability and prevents spurious generations.

The resulting benchmark thus consists of well-formed, unambiguous multi-hop tasks, each paired with a cached graph $\mathcal{G}$ and a minimal executable trajectory $P^\star$, enabling reproducible training and evaluation.

2.2 Benchmark Details

A key feature of our pipeline is that it is *fully automatic*: once a website graph is constructed, tasks can be generated without any manual authoring or intervention. This design implies that, in principle, the pipeline can scale to produce arbitrarily many tasks across arbitrarily many domains.

Domains. In this paper, we instantiate the benchmark on over 50 publicly accessible websites spanning healthcare, finance, e-commerce, entertainment, government, and scientific resources. This breadth ensures heterogeneous evaluation settings, as summarized in Tab. 4.

Number of Tasks. In total, we generate *5,000* intra-website tasks across the covered domains. Within the Healthcare and Finance group of websites, we generate *600* cross-website inter-website tasks. These generated tasks are used for experiments in this paper.

Pipeline. We release the generation pipeline to allow end users to create up-to-date tasks directly from live websites. This supports continual refresh of evaluation data, reduces overfitting to a static benchmark, and reframes **GTA** as a reusable task-generation framework rather than a one-time dataset.

3 Analysis

In this section, we first compare query structures across datasets (§3.1). We then quantitatively assess task difficulty by benchmarking performance against a simple search-based baseline, revealing that, unlike previous datasets, **GTA** tasks cannot be trivially solved by single-hop retrieval (§3.2). We perform a page-coverage analysis to measure how broadly each benchmark explores the underlying website graph, showing that **GTA** achieves substantially higher structural diversity (§3.3). We present human evaluation results confirming that **GTA** yields complex, verifiable, and realistic tasks for web-agent assessment (§3.4). Finally, we present a time and cost efficiency analysis (§3.5).

3.1 Qualitative Comparison of Queries

A closer examination of representative tasks highlights substantial differences between **GTA** and prior benchmarks. Unlike WebDS, which largely consists of factual lookups or pairwise comparisons (e.g., retrieving COVID-19 statistics or population growth rates), **GTA** tasks are explicitly compositional. They often require integrating multiple heterogeneous sources of evidence, such as combining numerical statistics (career strikeouts), product attributes (price and ratings), and domain-specific safety constraints (pharmacological dosage and drug interactions). This design ensures that tasks involve multi-hop reasoning rather than collapsing into atomic queries.

Another distinction lies in the nature of the required reasoning. **GTA** emphasizes *context-dependent decision-making*, where answers are not limited to factual retrieval but also demand comparative or evaluative judgments (e.g., identifying which product is superior, or whether a medical combination is safe). By contrast, AgentTrek and NNetNav focus on navigation-oriented instructions whose outcomes are often underspecified (e.g., "expand results on SeatGeek" or "find a restaurant").

Finally, the answer space of **GTA** tasks is designed to be precise and verifiable, grounded in structured numerical, categorical, or medical evidence. This stands in contrast to the open-ended or underspecified nature of prior benchmarks, where the lack of explicit answers limits their utility for rigorous evaluation. Collectively, these properties position **GTA** as a benchmark that emphasizes *long-horizon, evidence-grounded task construction*, thereby enabling a more faithful assessment of reasoning and integration capabilities in web agents.

3.2 Difficulty Comparison Against a Simple Search Baseline

We first evaluate a simple baseline where an LLM retrieves results from the Google Search API and generates an answer based on the returned snippets. To enable a fair comparison, we filter information-retrieval style tasks from both AgentTrek and NNetNav. As shown in Fig. 1, the correctness rate is surprisingly high: 95% on AgentTrek and 100% on NNetNav. On **GTA**, however, the baseline only reaches 14%. This indicates that tasks in AgentTrek and NNetNav often reduce to single-hop fact lookup, which can be trivially solved by a search engine. The inflated correctness rates reveal that these datasets do not capture the core challenge of web agents: reasoning over dispersed information across multiple pages and interfaces.

Dataset	Task	Answer / Outcome
GTA	Who has more career strikeouts in their MLB career, Derek Hill or Caleb Ferguson?	Caleb Ferguson has more career strikeouts (362).
	What is the total combined price of the UA Blur Pro Men's Football Cleats and the UA Spotlight Pro Men's Football Cleats, and which of these two products has the higher overall customer rating?	The total combined price is $240 ($110 for UA Blur Pro and $130 for UA Spotlight Pro). The UA Spotlight Pro Men's Football Cleats has the higher overall customer rating of 4.7 compared to 4.5 for the UA Blur Pro Men's Football Cleats.
	What is the maximum single dose of diphenhydramine (in mg) recommended for insomnia in adults, and is it safe to take this dose concurrently with sodium oxybate according to the interaction warnings?	The maximum single dose of diphenhydramine recommended for insomnia in adults is 76 mg (as diphenhydramine citrate), and it is not safe to take this dose concurrently with sodium oxybate due to potential additive CNS depression effects.
WebDS	Which currency pair (GBP/USD, EUR/JPY, USD/AUD) had the largest fluctuation in past 72h?	EUR/JPY had the largest fluctuation (high: 163.81 JPY).
	Compare 2020 population growth rates of Brazil vs. Indonesia.	Brazil: 0.72%. Indonesia: 1.07%. Indonesia higher.
	Total confirmed COVID-19 cases in California (2022).	5,634,397 cases.
AgentTrek	Explore Wikipedia's "Did you know?" section.	N/A
	Search for family events in Miami, FL and expand results on SeatGeek.	N/A
NNetNav	Find healthy dinner ideas.	N/A
	Find event spaces in San Francisco, CA.	N/A

Table 2: Example web tasks sampled from four datasets: GTA, WebDS, AgentTrek, and NNetNav. Each row shows a representative task and its summarized answer or outcome.

3.3 Page-Coverage Analysis

We next turn to the *page-coverage rate*, defined as the fraction of first-level nodes in the crawled website graph (roughly corresponding to distinct site functionalities, such as "team roster," "product details," or "checkout page") that are exercised by benchmark tasks. This metric is inspired by the *line-coverage* concept in programming languages, which measures how much of the codebase is actually exercised by a test suite. Analogously, high page coverage implies that benchmark tasks probe a diverse set of site functionalities, while low coverage indicates narrow task construction.

As shown in Fig. 1, the coverage is extremely limited: 20.2% for AgentTrek, 24.3% for NNetNav, and only 11.0% for WebDS, which are all far lower than **GTA** page coverage rate. This narrow span suggests that existing datasets focus disproportionately on a small subset of site functionality. The overlooked regions represent critical parts of the user experience, yet they remain underrepresented in benchmark tasks. Taken together, these analyses demonstrate that prior benchmarks (i) collapse to factoid-style queries easily solved by search engines, and (ii) induce strong exploration biases that leave large swaths of website functionality untested.

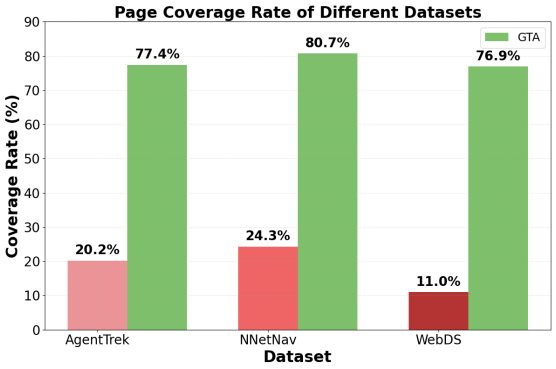

Figure 1: Page coverage rate comparison across datasets. Our proposed benchmark (GTA) achieves substantially higher page coverage (77.4%–80.7%) compared to prior datasets such as AgentTrek (20.2%), NNetNav (24.3%), and WebDS (11.0%). This indicates that GTA tasks exercise a much broader range of site functionalities, reflecting stronger information-sourcing ability. Statistics calculated on `espn.com` for AgentTrek, `underarmour.com` for NNetNav and `casper.com` for WebDS.

3.4 Human Evaluation

Correctness. To further assess the reliability of our automatically constructed benchmark, we randomly sample 100 tasks and evaluate the correctness of the annotated answers through human judgment. For each sampled task, we provide annotators with the generated task description, the proposed answer, and the two source web pages from which the answer is derived. Annotators

are asked to verify whether the annotated answer is indeed correct given the provided evidence. The resulting human agreement rate reaches 87%, demonstrating that our pipeline produces tasks and answers that are both interpretable and verifiable, with only a small fraction of cases requiring clarification.

Complexity, Utility, Realism and Quality. We further conduct a controlled human study (20 tasks per dataset) to compare tasks generated by our approach (**GTA**), WebDS, and AgentTrek along four dimensions: complexity, utility, realism, and overall quality. Results are shown in Table 3.

Dataset	Complexity	Utility	Realism	Quality
WEBDS	1.90	1.98	1.68	1.77
AGENTTREK	1.23	2.00	**1.98**	1.73
GTA	**2.35**	**2.03**	1.53	**1.82**

Table 3: Human evaluation (Likert 1–3) of task complexity, utility, realism, and overall quality. The highest value per column is bolded.

Annotators judge **GTA** tasks to be substantially more complex (**2.35**) than those in WebDS (1.90) and AgentTrek (1.23), indicating that our pipeline successfully produces multi-hop navigation and reasoning scenarios rather than trivial, single-hop lookups. On utility, **GTA** achieves the highest rating (**2.03**), followed closely by AgentTrek (2.00) and WebDS (1.98), suggesting that **GTA** maintains comparable usefulness with added reasoning depth.

For realism, AgentTrek scores the highest (**1.98**), outperforming WebDS (1.68) and **GTA** (1.53). This gap may stem from our graph-grounded generation process, which can introduce queries that are less reflective of natural user intents. Finally, **GTA** attains the best overall quality score (**1.82**), surpassing both WebDS (1.77) and AgentTrek (1.73), highlighting that tasks generated by our approach are both coherent and challenging.

Taken together, these results demonstrate that **GTA** yields tasks that best balance realism, reasoning depth, and evaluation value—producing scenarios that are both practically useful and more faithful to the complexities of real-world web navigation.

3.5 Time and Cost Efficiency

A key advantage of **GTA** lies in its scalability and efficiency. We present detailed quantitative comparisons in Appx. §D. In summary, **GTA** achieves a dramatic reduction in both computational and financial overhead compared to prior task-generation pipelines.

4 Experiments

In this section, we first introduce the baseline agents evaluated in this paper (§4.1). We then report their performance on **GTA** intra- and inter-website tasks (§4.2). We also extend evaluation to multilingual web environments (§4.3). Finally, we conduct an error analysis to identify key failure modes (§4.4). Specific implementation details are presented in Appx. §G.

4.1 Baselines

Browser Use.[2] Browser Use is an open-source framework that couples large language models with a real browser environment. The perceive–decide–act cycle enables robust handling of multi-step workflows such as navigation, form-filling, and information extraction across diverse sites. It is a widely adopted baseline, achieving strong results on benchmarks like WebVoyager.

AgentOccam. (Yang et al., 2025) AgentOccam represents a complementary approach that prioritizes simplicity and alignment. It refines the interface between the LLM and the web environment by pruning rarely used actions and filtering page observations into concise textual representations. It has achieved state-of-the art performance on benchmarks like WebArena.

We evaluate a limited number of agent backbones by design, consistent with prior web-agent benchmarks, which typically focus on representative agent paradigms rather than exhaustive architectural sweeps, for example, WebDS(Hsu et al., 2025) evaluates Browser Use and AgentOccam, while NNetNav(Murty et al., 2025) and WebWalker(Wu et al., 2025) primarily study ReAct-style agents. Our goal is to validate the benchmark under commonly used agent behaviors; accordingly, the primary contribution lies in the benchmark itself rather than in the breadth of agent architectures evaluated.

4.2 Benchmark Results

Intra-website Task Performance. Fig. 2 reports success rates of Browser Use and Agen-

[2] https://github.com/browser-use/browser-use.

tOccam across domains. A clear gap emerges between our benchmark and prior ones such as WebVoyager and Mind2Web. While both agents achieve relatively high scores on these earlier datasets (up to 82% on WebVoyager and 45% on Mind2Web), their performance on our benchmark domains is markedly lower, often below 20–30%. This contrast suggests that existing benchmarks have become static and saturated, allowing agents to overfit to templated interfaces or repeated patterns, whereas our benchmark introduces dynamic, evolving sites that demand genuine adaptation and remain useful over a longer horizon.

Beyond the overall drop in success rate, we also observe sharp performance variations across domains. As shown in Fig. 8 `musicbrainz.org` or `cvshealth.com` admit moderate success, whereas shopping portals like `underarmour.com` and `ticketcenter.com` prove substantially harder. Such heterogeneity highlights that real-world navigation cannot be captured by single-domain evaluation: robustness requires agents to generalize across diverse layouts, workflows, and interaction styles.

Inter-website Task Performance. To demonstrate the flexibility of our pipeline, we also generate *inter-website multi-hop tasks*, where solving a task requires integrating information from multiple websites rather than staying within a single domain. For these experiments, we select two domains with naturally complementary sources of information: healthcare (Mayo Clinic, WebMD, Drugs.com, CVS Health, and CDC) and finance (Yahoo Finance, markets.ft and Seeking Alpha).

We evaluate these tasks using the Browser Use agent, which executes tasks in a real browser environment. Experiments show that success rates for cross-website tasks are substantially lower than those observed in single-website settings. For example, Mayo Clinic and WebMD tasks achieve a 12.5% success rate, CDC and WebMD reach 8.2%, and Drugs.com & CVS Health drop to just 4.2%. These numbers are considerably lower than the success rate of same-website tasks reported earlier, confirming that cross-website integration poses unique challenges.

We attribute this difficulty to two factors. First, cross-website tasks require agents to reason about *complementary but non-overlapping knowledge*: one site may provide background definitions or guidelines, while the other presents instance-specific details (e.g., drug side effects vs. dosage instructions). Second, agents must manage *distributional shifts across sites*, such as inconsistent terminology, different UI structures, or divergent information formats. This stands in contrast to same-site navigation, where once an agent adapts to the local interface, it can generalize within that site.

4.3 Multi-lingual Task Performance

To further evaluate the flexibility and coverage of our pipeline, we extend task generation beyond English to *multilingual web environments* spanning the *health*, *sports*, and *news* domains. We crawled representative websites in four languages—English, Italian, German, and Japanese—and constructed tasks from each. We then evaluate these tasks using two representative web agents, Browser Use and AgentOccam. As shown in Fig. 9, the results reveal a pronounced degradation in performance on non-English websites. For example, the Browser Use agent attains a 0% success rate on the German site, 2.9% on the Japanese site, and 0% on the Italian site. In contrast, the same agent achieves markedly higher performance on English-language health websites, with success rates of 8.2% on CDC and 6.0% on NIH.gov. These findings highlight a strong bias in current web agents toward English-centric environments, while exposing severe limitations in their ability to operate on foreign-language websites. A similar trend is consistently observed with AgentOccam and across other domains, corroborating the language-specific performance disparity.

4.4 Error Analysis

We conduct a detailed error analysis on a random sample of 100 failed tasks to better understand the limitations of current web agents on our benchmark. Three major categories of failure modes emerge.

Failure to Reach All Required Webpages (90%). A common error pattern arises when the agent fails to correctly navigate to all webpages required by a multi-hop query. Since many tasks require synthesizing evidence from two or more distinct sites, failure to reach even one of them leads to incomplete reasoning chains and ultimately in-

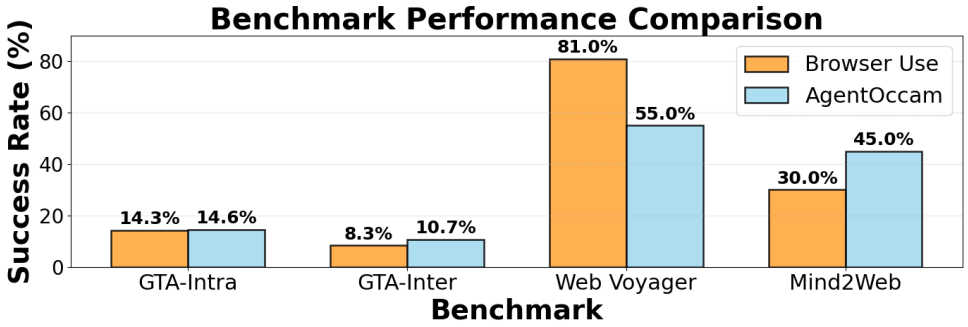

Figure 2: Benchmark performance comparison of Browser Use and AgentOccam. On GTA (intra/inter-website) tasks, both agents achieve relatively low success rates, highlighting the difficulty of multi-hop, compositional reasoning. By contrast, performance is saturated on WebVoyager and Mind2Web. Human performance on GTA tasks is 85%.

correct answers. We observe that this issue often stems from brittle link-following strategies, where the agent either misinterprets hyperlink semantics (e.g., clicking on visually similar but contextually irrelevant links) or becomes trapped in navigation loops.

Early Stopping (40%). Another prominent failure mode involves premature termination of the task. Agents sometimes generate a final answer after retrieving partial evidence, without fully verifying consistency across all required sources. This behavior is especially prevalent in tasks that involve numerical aggregation or comparisons, where one piece of evidence might seem sufficient but is actually incomplete. Such early stopping reflects the broader challenge of balancing efficiency with completeness in long-horizon reasoning.

Over-reliance on Search Box Operations (40%). We also find that agents frequently overuse on-site search boxes rather than exploiting the explicit structure of the web graph. While search boxes sometimes provide shortcuts, they are not guaranteed to expose the necessary evidence, and reliance on them can bypass important compositional navigation steps. In many cases, agents retrieve irrelevant or overly broad results through poorly formulated queries, which then propagate errors into the reasoning process.

5 Related Work

A number of benchmarks have been developed to evaluate web agents. Mind2Web (Deng et al., 2023) provides human-authored task–solution pairs for open-domain web navigation. WebVoyager (Wang et al., 2023), WebArena (Zhou et al., 2024), BrowseComp (Wei et al., 2025) and WebDS (Hsu et al., 2025) provide realistic environments and curated tasks for web-based decision making. However, these benchmarks mainly provide end-to-end trajectories, offering limited support for fine-grained diagnostics and compositional reasoning.

To reduce annotation costs, recent work has explored automatic task generation. AgentTrek (Xu et al., 2024) transforms web tutorials into executable trajectories, while WebWalker (Wu et al., 2025) and NNetNav (Murty et al., 2025) use LLM-driven exploration to synthesize tasks from raw environments. Despite their promise, these approaches can be expensive due to repeated rollouts, biased toward salient regions of a site, and often produce tasks that collapse into simple single-hop retrieval. **GTA** also differs in construction strategy. Unlike WebWalker, which discovers tasks through model-driven exploration, **GTA** first systematically crawls websites, builds explicit site graphs, and then generates tasks from enumerated graph structures. This design reduces exploration bias and enables controlled graph coverage and cross-page evidence composition. Compared with human-authored benchmarks such as BrowseComp, **GTA** is fully automated and scalable across 50+ websites, while explicitly guaranteeing multi-hop evidence distribution, cross-page dependency, and non-answerability from any single page.

6 Conclusion

We introduced **GTA**, a scalable benchmark for web agents. By decoupling crawling from task generation and anchoring tasks to the site graph, **GTA** produces compositional, verifiable tasks with executable trajectories across diverse domains and languages.

Our analyses show that **GTA** surfaces challenges overlooked by prior benchmarks—agents fail on cross-website and multilingual tasks, and simple search baselines cannot solve them. Human studies further confirm that tasks are more complex, useful, and diagnostic. We hope **GTA** will serve as a foundation for advancing robust, generalizable web agents.

Limitations

Scope of Website Coverage. Although GTA spans over 50 publicly accessible websites across diverse domains and languages, the current selection remains limited relative to the vast heterogeneity of the modern web. Certain interaction types—such as authentication-gated services, dynamically rendered JavaScript interfaces, and transaction-based workflows—are excluded due to ethical and security constraints. Extending GTA to such settings will require robust sandboxing and privacy-preserving mechanisms for exploration.

Quality Verification and Reasoning Depth. The automatic verification process, though effective at filtering low-quality or ambiguous tasks, still depends on LLM-based validators. Subtle reasoning errors or underspecified compositional dependencies may occasionally persist. Incorporating human-in-the-loop validation or multi-model consensus could further strengthen quality assurance, especially for tasks involving nuanced domain knowledge.

Dependence on Crawl Snapshots. Our pipeline relies on periodic crawls to construct site graphs, enabling scalable data collection and timely iteration. However, task difficulty may vary due to multiple factors, including structural updates in websites (Ishmam and Marino, 2026), the time interval between task creation and evaluation, and content drift or removal. These dynamics can affect task solvability and consistency across different crawl snapshots. Future work should explore more generalizable web agents that are robust to such temporal and structural variations.

Scope of Task Types. **GTA** focuses on multi-hop information-seeking tasks rather than the full range of web agent behaviors. In particular, it does not cover procedural workflows such as form-filling, checkout, or other action-heavy interactions. This scope is intentional: prior information-seeking benchmarks often collapse into single-hop retrieval, leaving cross-page evidence integration underexplored. Procedural workflows are an important complementary challenge and are already represented in benchmarks such as WebArena.

Ethics Statement

We strictly adhere to ethical and legal standards in data collection and experimental design. All web data used in this study were obtained through automated scripts that fully comply with each website's crawler and robots.txt policies. We only accessed publicly available webpages and refrained from interacting with or collecting any user-specific or private information. The crawling process respected server load limitations by introducing randomized delays and rate-limiting to minimize network impact.

Additionally, all data were used solely for research purposes and are released or analyzed in aggregate form to prevent any potential misuse or privacy risk. Our benchmark construction and experiments comply with institutional and community ethical guidelines for web data research.

Acknowledgments

Muhao Chen was supported in part by the National Science Foundation under grants OAC-2531126 and ITE-2333736.

References

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Ethan Hsu, Hong Meng Yam, Ines Bouissou, Aaron Murali John, Raj Thota, Josh Koe, Vivek Sarath Putta, G K Dharesan, Alexander Spangher, Shikhar Murty, Tenghao Huang, and Christopher D. Manning. 2025. Webds: An end-to-end benchmark for web-based data science.

Tenghao Huang, Kinjal Basu, Ibrahim Abdelaziz, Pavan Kapanipathi, Jonathan May, and Muhao Chen. 2025a. R2D2: Remembering, replaying and dynamic decision making with a reflective agentic memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30318–30330, Vienna, Austria. Association for Computational Linguistics.

Tenghao Huang, Sihao Chen, Muhao Chen, Jonathan May, Longqi Yang, Mengting Wan, and Pei Zhou.

2025b. Teaching language models to gather information proactively. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 15588–15599.

Tenghao Huang, Dong Hee Lee, John Sweeney, Jiatong Shi, Emily Steliotes, Matthew Lange, Jonathan May, and Muhao Chen. 2025c. Foodpuzzle: Toward developing large language model agents as autonomous flavor scientists. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, page 5493–5504, New York, NY, USA. Association for Computing Machinery.

Md Farhan Ishmam and Kenneth Marino. 2026. Timewarp: Evaluating web agents by revisiting the past. *arXiv preprint arXiv:2603.04949*.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher Manning. 2025. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144. PMLR.

Paloma Sodhi, S.R.K Branavan, Yoav Artzi, and Ryan McDonald. 2024. Step: Stacked LLM policies for web actions. In *First Conference on Language Modeling*.

Alexander Spangher, Tenghao Huang, Philippe Laban, and Nanyun Peng. 2025. Creative planning with language models: Practice, evaluation and applications. In *Proceedings of the 2025 Annual Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 5: Tutorial Abstracts)*, pages 1–9, Albuquerque, New Mexico. Association for Computational Linguistics.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi (Jim) Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, et al. 2025. Webwalker: Benchmarking llms in web traversal. *arXiv preprint arXiv:2501.07572*.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. 2024. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2025. Agentoccam: A simple yet strong baseline for LLM-based web agents. In *The Thirteenth International Conference on Learning Representations*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2024. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

A Website Coverage

Tab. 4 shows representative websites covered in **GTA** benchmark.

B Prompt Details

Fig. 3 shows detailed prompt of multi-hop web agent task creation. Fig. 4, Fig. 7, and Fig. 5 show detailed prompts of the quality control pipeline.

C Performance Per Website

Fig. 8 shows the success rates of BROWSERUSE and AGENTOCCAM on a subset of websites.

D Cost of Task Generation

Computational Cost. An important consideration in designing **GTA** is the computational and monetary cost of generating large-scale tasks. Our pipeline is deliberately lightweight: the only substantive expense lies in producing webpage descriptions for retrieval indexing. Crawling itself incurs no cost, and in-context task generation requires only a small number of LLM calls.

Concretely, generating descriptions for approximately 2,000 webpages and generating 100 multi-hop tasks via in-context prompting costs around $10. Once the site graph $\mathcal{G}$ is constructed and indexed, however, the marginal cost of additional tasks is effectively negligible: the graph enables arbitrarily many tasks to be generated without re-incurring indexing costs. By contrast, prior benchmarks such as AGENTTREK report an average cost of $0.55 per task, as they rely on LLM-based exploration policies to traverse websites. This design makes the marginal cost of each new task significantly higher and limits scalability, whereas our graph-based approach amortizes indexing cost and supports large-scale task generation at roughly $0.10 per task in our experiments. This low overhead is particularly advantageous for dynamic test sets, where new tasks can be generated continuously with minimal additional expense.

Time Cost. In addition to monetary efficiency, our pipeline is designed to be lightweight in terms of time. The most time-consuming stage is crawling webpages and creating retrieval index. For a medium-scale setting of approximately 2,000 webpages, this process completes within 30–40 minutes on a single machine. Once the site graph is built, task generation itself is relatively fast, allowing us to rapidly scale the benchmark to thousands of multi-hop tasks without significant runtime bottlenecks. This property stands in contrast to LLM-exploration-based benchmarks, where each new task requires running an agent rollout through the website, incurring substantial time costs per task.

E Multilingual Task Performances

Fig. 9 shows web agents' cross-lingual performances. Both BROWSERUSE and AGENTOCCAM exhibit a consistent decline in success rate when moving from English to non-English sites. The results highlight a strong bias in current web agents toward English-centric environments, while exposing severe limitations in their ability to operate on foreign-language websites.

F Annotation Details

We recruit 3 annotators with computer science background for our human evaluation. An example annotation interface is shown in Fig. 10. The inter-annotator agreements (Cohen's Kappa) is 0.3, which indicates fair agreement.

G Implementation Details

We use GPT-4o as the underlying large language model for agent exploration and GPT-5.1 for task generation. Our web interaction stack is built on Chromium, which is used to render webpages and capture visual snapshots for agent perception. This setup matches that of prior web-agent benchmarks, enabling consistent state replay and fair comparison. Each environment step corresponds to a deterministic browser action followed by a rendered page snapshot.

We inherit environment settings and site configurations from established benchmarks where applicable (e.g., using the same website instances and task distributions as WebArena and WebDS). To ensure safety and reproducibility, we restrict the agent to publicly accessible webpages and explicitly avoid login-gated content, personal accounts, or sensitive transactional flows.

Dynamic webpage elements such as pop-ups, cookie consent banners, and modal dialogs are handled using the same rules and heuristics defined in the corresponding benchmark environments. We ensures that agent behavior remains comparable to prior baselines and that observed improvements stem from reasoning and decision-making rather than environment customization.

Domain	Websites
Healthcare and Medicine	nih.gov, cdc.gov, mayoclinic.org, webmd.com, drugs.com, pubmed.ncbi.nlm.nih.gov, cvshealth.com, mhlw.go.jp, salute.gov.it, nhc.gov.cn, health.people.com.cn, bundesgesundheitsministerium.de, sante.gouv.fr
Finance and Economics	bloomberg.com, finance.yahoo.com, markets.ft.com, seekingalpha.com, morningstar.com, bea.gov, stlouisfed.org, fred.stlouisfed.org, tradingeconomics.com, consumerfinance.gov, sec.gov
E-commerce and Consumer Platforms	underarmour.com, casper.com, resy.com, carnival.com, expedia.com, yellowpages.com
News, Entertainment, and Social Media	espn.com, pro-football-reference.com, fbref.com, last.fm, musicbrainz.org, ticketcenter.com, tripadvisor.com, facebook.com
Government, Science, and Public Resources	noaa.gov, climate.gov, nps.gov, mbta.com, new.mta.info, dmv.virginia.gov, iata.org, worldpop.org, ourworldindata.org, statista.com, riaa.com, ir.mit.edu

Table 4: Domain coverage of our benchmark. Sites span healthcare, finance, e-commerce, entertainment, and government/science, ensuring diverse and multilingual evaluation settings.

You are a helpful assistant that creates multi-hop web navigation tasks for an AI agent. Given two different websites, create a challenging question that requires the agent to:

1. Extract information from the first website.
2. Use that information to find and navigate to the second website.
3. Extract additional information from the second website.
4. Combine information from both websites to answer the question.

Input Websites:
Website 1: <web_1>
Website 2: <web_2>

Task Requirements:

- Must require information from **both websites**.
- Should test the agent's ability to connect information across sites.
- The answer must be deterministic (no ambiguity or open-endedness).
- Do not explicitly mention "Website 1" or "Website 2" in the question.
- The question should be a natural combination of the two sources.

Output Format:

- `TASK:` [The multi-hop question]
- `RATIONALE:` [Why both websites are needed]
- `ANSWER:` [Final answer, no placeholders]

Example:
`TASK:` What is the final score of the Lakers game mentioned on the first website, and which player from that game has the highest career scoring average according to their player profile on the second website?
`RATIONALE:` This requires extracting game information from the first site, then navigating to player profiles on the second site to find career statistics.
`ANSWER:` [Provide actual answer here]

Figure 3: Prompt template for generating multi-hop web navigation tasks.

You are an expert at analyzing query ambiguity. Please assess whether the following query is ambiguous or clear.
QUERY TO ANALYZE: `{query}`

Assessment Criteria. Ambiguous if:

1. Contains temporal questions without a time period.
2. Uses vague time references like "recently" or "lately".
3. Could refer to multiple possible periods or events.
4. Lacks sufficient context to identify the specific instance.

Clear if:

1. Includes explicit temporal context (year, month, season, etc.).
2. Provides enough context to identify the event or period.
3. Contains unambiguous references that can be definitively answered.

Examples. - Ambiguous: "`When did the team last win a championship?`" - Clear: "`When did the Lakers last win an NBA championship?`"

Figure 4: Prompt for ambiguity assessment of queries.

You are an expert at analyzing query structure. Please assess whether the following query is a valid multi-hop query or simply a concatenation of unrelated queries.
QUERY TO ANALYZE: `{query}`

Assessment Criteria. Concatenated if:

1. Two unrelated questions joined by "and".
2. Each can be answered independently.
3. No logical connection between parts.
4. Different entities/topics with no relationship.

Valid Multi-hop if:

1. Questions are related and share context.
2. One part helps answer the other.
3. Logical connection exists between the two.
4. Entities or events are related.

Examples. - Concatenated: "`Who is the starting pitcher for the Giants, and what is the link to the depth chart?`" - Valid Multi-hop: "`On which dates did both Ben Rice and Randy Arozarena hit home runs, and how many days apart were the two events?`"

Figure 5: Prompt for distinguishing concatenated vs valid multi-hop queries.

You are an expert at fixing ambiguous queries by adding specific context from web content.
Original Ambiguous Query: <original_query>
Available Web Content for Context:
<web_context_text>

Instructions. Fix the ambiguous query by adding specific temporal or contextual information from the web content. The clarified query should be clear and unambiguous, while preserving the original intent.

Output Format. `{"FIXED_QUERY": "...", "CHANGES_MADE": "..."}`

Figure 6: Prompt template for fixing ambiguous queries using contextual information.

You are tasked with verifying the correctness of an answer based on website content.
Given:

- Website_1 content: <web_1>
- Website_2 content: <web_2>
- Question: <question>
- Provided answer: <answer>

Consider.

1. Is the answer factually accurate according to the website content?
2. Is the answer complete and does it address the question?
3. Are there contradictions with the website content?

Output Format. Respond with only `Yes` if the answer is correct and supported, or `No` if incorrect, unsupported, or contradictory.

Figure 7: Prompt template for verifying correctness of answers against website content.

Figure 8: Performance comparison of Browser Use and AgentOccam across representative websites. Each bar shows the average success rate (%) for completing web-based tasks within a given domain.

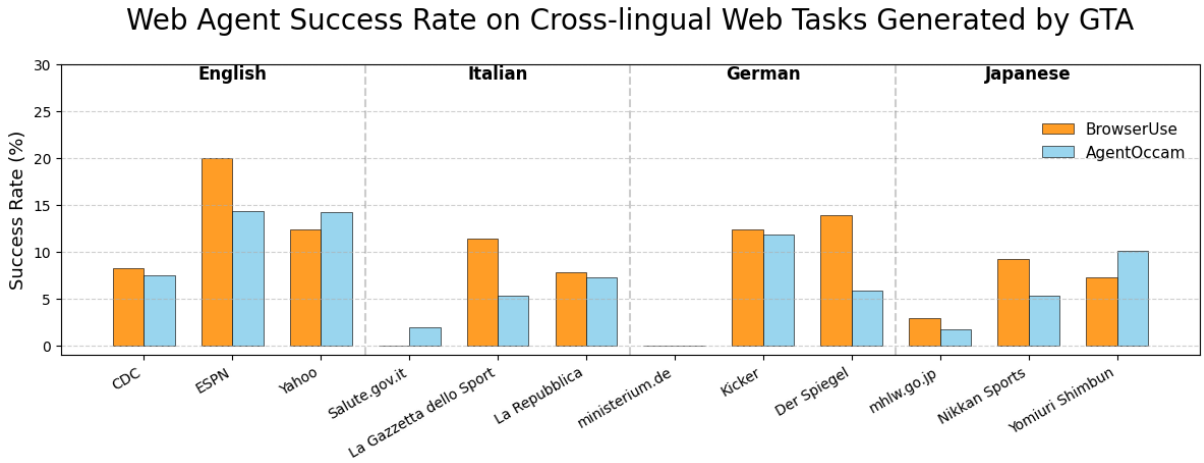

Figure 9: Cross-lingual performance of BrowserUse and AgentOccam performances on web tasks generated by GTA. Each group of bars represents websites in English, Italian, German, and Japanese domains. The overall decline in success rate across languages highlights the growing challenge of multilingual generalization for web agents, especially when adapting to heterogeneous site structures and contents.

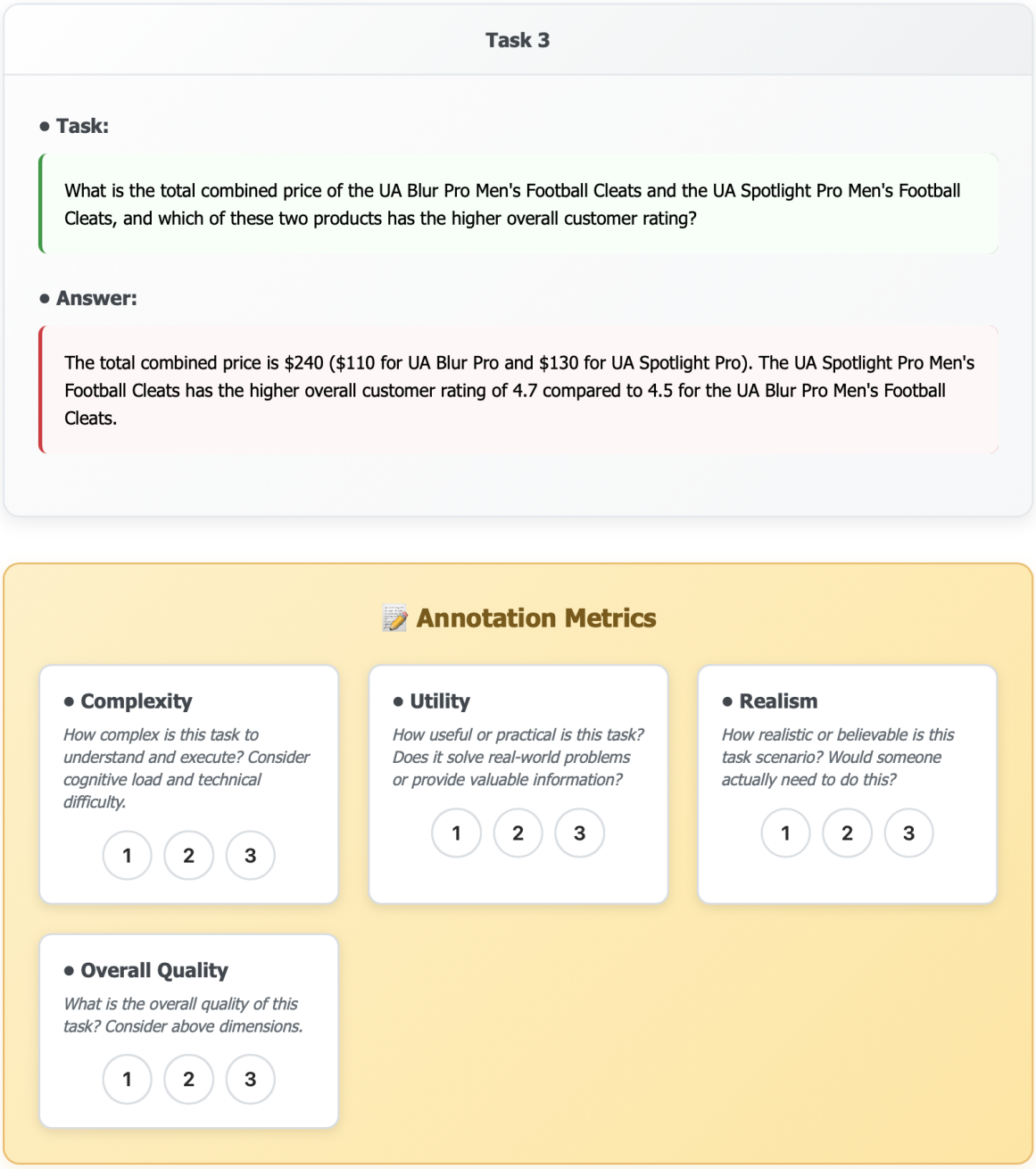

Figure 10: Example annotation interface