BaRA: BFS-and-Reflection Web Data Collection Agent

Anonymous ACL submission

Abstract

Large language model (LLM)-based web agents reduce manual scripting for web data collection, yet on live websites, they often miss relevant pages, return incomplete multimodal outputs, or return media URLs that are not directly downloadable. We present **B**FS-**a**nd-**R**eflection **A**gent (BaRA), a framework for site-level collection under a fixed interaction budget. The framework combines bounded breadth-first search (BFS) traversal with history-based self-reflection. We evaluate BaRA on 50 synthetic websites with ground-truth reference sets. We additionally test on three public websites with cluttered or dynamic layouts. BaRA outperforms Pure LLM, SeeAct-Vision, and Browser-use on link discovery and downloadable multimodal extraction, with the largest gains in download-valid image and video recovery. Our code is available at https://anonymous.4open.science/r/BaRA-9032.

1 Introduction

Web data collection remains a practical bottleneck for natural language processing (NLP) and multimodal pipelines, including dataset construction, information extraction, and retrieval-augmented generation. Prior work on web and graphical user interface (GUI) interaction has progressed from controlled environments such as World of Bits and MiniWoB++ (Shi et al., 2017; Liu et al., 2018) and simulated settings such as WebShop (Yao et al., 2022a) to large-scale benchmarks on real websites, including Mind2Web, WebArena, VisualWebArena, WorkArena, WorkArena++, WebLINX, and Online-Mind2Web (Deng et al., 2023; Zhou et al., 2023; Koh et al., 2024; Drouin et al., 2024; Boisvert et al., 2024; Lù et al., 2024; Xue et al., 2025). Recent large language model (LLM)-based agents have made live website interaction more accessible (Nakano et al., 2021; He et al., 2024; Zheng et al., 2024; Furuta et al., 2023; Zheng et al., 2023; Yang et al., 2024). However, robust crawling and extraction in the wild remain fragile: dynamic interfaces, heterogeneous layouts, and execution failures frequently yield partial results, and gains observed on controlled benchmarks often weaken under realistic online evaluation (Pan et al., 2024; Xue et al., 2025). One key challenge is that these failures are often *silent*: agents terminate without explicit errors while still missing relevant pages or modalities, and the extracted media URLs are often non-downloadable or hallucinated.

We study a practical but under-emphasized setting: *site-level data collection under a fixed interaction budget*. Given a seed URL, the goal is to cover a bounded set of internal pages and return verifiable multimodal artifacts with provenance. This differs from standard single-task web navigation benchmarks because coverage and artifact validity matter as much as task completion.

We propose the BFS-and-Reflection Agent (BaRA), a control layer for site-level web data collection on top of an existing browser-executed runtime. BaRA addresses two common failure modes: premature commitment to a single deep branch and brittle failure after incomplete runs. To do so, it combines bounded breadth-first search (BFS) traversal for broader site coverage with history-based self-reflection that revises task instructions after unsuccessful attempts. The pipeline further applies URL normalization, deduplication, and download validation before returning outputs.

Contributions. Our contributions are threefold: (i) a robustness-oriented collection framework that integrates bounded BFS traversal with history-based self-reflection under fixed interaction budgets; (ii) a constrained per-page extraction and validation protocol for returning machine-processable multimodal artifacts; and (iii) an evaluation setup that separates link discovery from download-valid extraction across synthetic and live web settings.

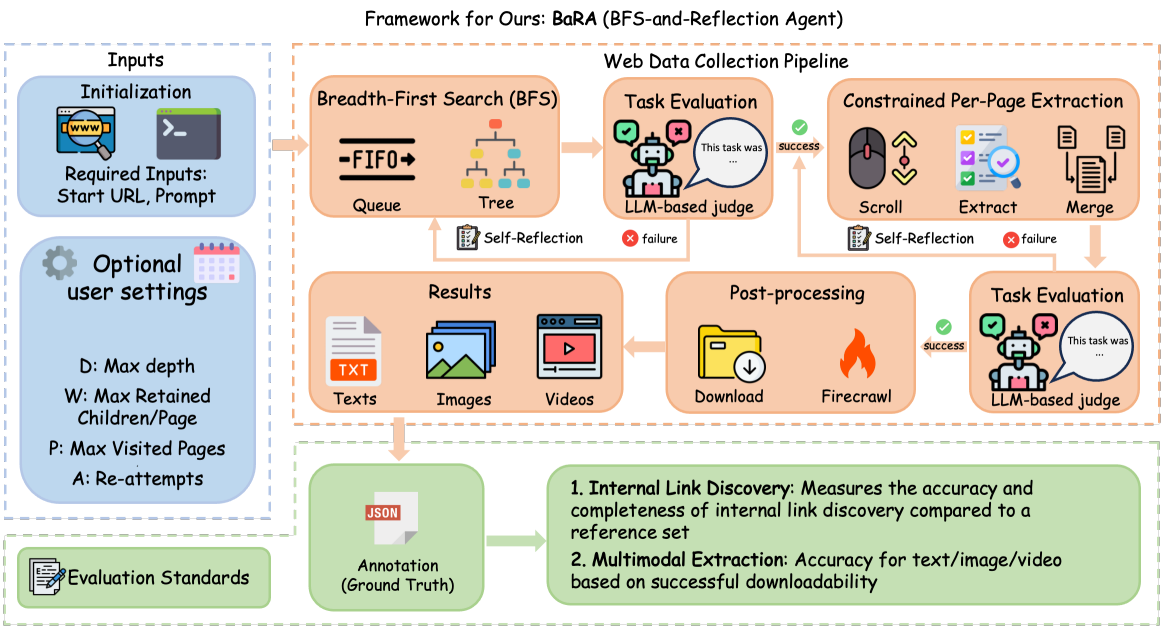

Figure 1: Overview of BaRA. Starting from a seed URL and user-configurable budgets (maximum BFS depth D, maximum retained child links per page W, maximum visited pages P, and maximum number of retry attempts A), the system first performs bounded BFS traversal, then applies constrained per-page extraction to each discovered page, and finally validates and downloads the recovered media. When exploration or extraction fails, a history-based self-reflection module rewrites the prompt for the next attempt. The pipeline returns download-valid artifacts together with per-page metadata for evaluation.

2 Related Work

Web-agent benchmarks and environments. Web-agent research has progressed from controlled browser environments such as World of Bits and MiniWoB++ (Shi et al., 2017; Liu et al., 2018) to more realistic and scalable settings. WebShop introduced a simulated e-commerce environment with real products and compositional instructions (Yao et al., 2022a), while Mind2Web established a large-scale benchmark on real-world websites (Deng et al., 2023). Subsequent work broadened this direction through realistic autonomous interaction benchmarks such as WebArena and VisualWebArena (Zhou et al., 2023; Koh et al., 2024), enterprise-style knowledge-work benchmarks such as WorkArena and WorkArena++ (Drouin et al., 2024; Boisvert et al., 2024), and conversational navigation datasets such as WebLINX (Lù et al., 2024). Online-Mind2Web further highlights robustness under live web conditions and the remaining gap between benchmark progress and in-the-wild performance (Xue et al., 2025).

Web-agent methods, reasoning, and reflection. On the methods side, WebGPT improved evidence-grounded long-form QA through browser interaction (Nakano et al., 2021). More recent systems, such as WebVoyager and SeeAct-Vision, have further advanced live website interaction with multimodal grounding (He et al., 2024; Zheng et al., 2024). Other approaches broaden the design space through prompting, memory, or simplicity-oriented formulations: WebGUM studies instruction-finetuned multimodal navigation agents (Furuta et al., 2023), Synapse uses trajectory-as-exemplar prompting with memory for computer control (Zheng et al., 2023), and AgentOccam revisits design simplicity as a strong LLM-agent baseline (Yang et al., 2024). These lines of work are also closely related to reasoning and self-improvement paradigms such as ReAct and Reflexion (Yao et al., 2022b; Shinn et al., 2023). We build on these paradigms by adapting iterative reasoning and self-improvement to site-level web data collection under a fixed interaction budget, where bounded BFS traversal, constrained per-page extraction, and history-based self-reflection serve as central design choices.

Platforms and positioning. At the systems level, our implementation is built on top of Browser-use (Browser-use contributors, n.d.) and is conceptually related to broader infrastructure efforts such as BrowserGym, which aims to standardize envi-

ronments and evaluation for web agents (Chezelles et al., 2024). Accordingly, we position BaRA as a control layer for site-level web data collection on live websites under a fixed interaction budget. In this sense, our method complements existing runtimes and environments with bounded BFS traversal, constrained per-page extraction, history-based self-reflection, and download validation.

3 BaRA Agent: Core Method

This section details the bounded BFS traversal and history-based self-reflection components shown in Figure 1. BaRA is built around two design choices that directly target robustness under fixed interaction budgets: bounded BFS traversal for predictable link coverage and history-based self-reflection for failed or incomplete runs.

3.1 Problem Setup

Given a start URL u_0 (and optionally a user intent), BaRA collects same-site content under a fixed interaction budget. We denote the maximum BFS depth, maximum retained child links per page, maximum visited pages, and maximum retry attempts by D, W, P, and A, respectively. Let $U = (u_0, \ldots, u_{m-1})$ be the visited URL sequence with $m \leq P$. For a visited page u_i, let $C(u_i)$ denote its normalized and filtered set of candidate internal links, and let $\hat{C}(u_i) \subseteq C(u_i)$ denote the first at most W unique children retained for BFS expansion. The next two subsections describe BaRA's two core components: bounded BFS traversal and history-based self-reflection.

3.2 Coverage-Oriented Bounded BFS Traversal

Prompt-driven web agents often suffer from *direction loss*: once the agent commits to a promising path, it may over-exploit a single branch and miss large portions of the site. BaRA addresses this in Step 1 link discovery with bounded BFS traversal.

BFS controller. Starting from the start URL u_0 at depth 0, the agent maintains a first-in, first-out (FIFO) queue whose entries consist of a URL and its current depth. In addition, the controller maintains a global visited set over normalized URLs to prevent revisiting previously explored pages. At each step, the agent removes one entry from the queue, loads the corresponding page, and extracts candidate hyperlinks. It then (1) normalizes candidate URLs into a canonical form and removes page-level duplicates, (2) filters out off-site links and non-content endpoints (e.g., login or terms pages), (3) keeps at most the first W unique child links in Document Object Model (DOM) order, and (4) appends each retained child to the queue with depth increased by one only if the resulting depth does not exceed D, the normalized child URL is not already in the visited set, and it is not already present in the queue. All deduplication and membership checks are performed after URL normalization. The process ends when P pages have been visited, or the queue becomes empty.

Why BFS under fixed budgets. Bounded BFS provides two practical benefits in live web settings: (1) *predictable coverage*, as the agent systematically expands the frontier rather than drifting into deep paths; and (2) *traceability*, as the traversal yields an interpretable visitation record (`visited_order` and `by_depth`) that are inspected for debugging and analysis. The concrete Step 1 prompt that instantiates this bounded BFS traversal is reported in Appendix B.1.

3.3 History-Based Self-Reflection

Even with structured traversal, web environments remain brittle: dynamic DOM changes, pop-ups, rate limits, and timeouts often stall or return empty outputs. Rather than treating such failures as terminal, BaRA applies a history-based self-reflection loop that turns failed or incomplete runs into revised prompts for subsequent attempts.

Reflection signal. After each attempt, the system collects a compact summary of the run, including (i) the agent's execution history and (ii) the extracted content produced so far (possibly empty). A lightweight success evaluator determines whether the attempt produced *any usable signal* (e.g., non-empty text or media URLs). If successful, the system proceeds; otherwise, it triggers reflection.

Reflection-to-repair. When an attempt fails, the history-based self-reflection module uses the agent's execution history and partial outputs to produce a structured diagnosis: failure type, failure point, and prompt-level fixes. It then rewrites the next instruction:

$$p_{t+1} \leftarrow R(p_t, h_t, y_t).$$

where p_t is the instruction at attempt t, h_t is the execution history up to attempt t, y_t is the extracted output from attempt t, and $R(\cdot)$ is the reflection

module. For example, if a page loads but the agent returns empty media URLs after partial scrolling, R revises the next instruction to stay on the same URL, continue scrolling, and re-extract the media URLs. The agent retries under the revised instruction up to A times.

Scope of reflection. We apply the same recovery logic to both Step 1 (link discovery) and Steps 2–3 (page-level extraction and download validation): (1) *search-time failures* (e.g., the BFS stage fails to retrieve links on some pages) and (2) *extraction-time failures* (e.g., a page loads but the output is empty or malformed). In both cases, reflection aims to prevent the agent from *giving up* by producing a next-step instruction that is better aligned with the encountered website behavior. The corresponding failure-analysis and success-evaluation prompts are reported in Appendix B.2 and B.3.

4 Experimental Setup

We evaluate BaRA as a practical pipeline for *robust web data collection on live websites* under fixed interaction budgets. Our evaluation focuses on two end-to-end capabilities for downstream dataset construction: *(i) link discovery* (Step 1) and *(ii) downloadable multimodal extraction* (Steps 2–3).

What is being evaluated. Given a start URL, the system first produces a BFS-style traversal log (`visited_order` and `by_depth`) under explicit depth/width/page caps. Then, for each visited page, it performs Step 2 (page-level extraction): under the constrained per-page extraction protocol, the agent must stay on the target URL, scroll to reveal lazy-loaded content under a hard cap, and output raw text along with image and video URLs in a strict format. The concrete Step 2 single-page extraction prompt is given in Appendix B.4.

This decomposition serves the evaluation protocol. Step 1 measures whether the agent recovers the correct internal page URLs under fixed depth, width, and page budgets. Steps 2–3 then measure whether the pipeline turns those visited pages into raw text and download-valid media artifacts under the constrained per-page extraction protocol. These two stages fail in different ways: the traversal stage loses coverage when the agent drifts or exhausts the budget, whereas the extraction stage loses artifacts when the agent reaches the correct page but returns incomplete text, missing media URLs, or non-downloadable outputs. Evaluating both stages explicitly lets us separate coverage errors from artifact-level errors across both synthetic and real-world settings, rather than conflating failures in traversal with failures in extraction.

4.1 Benchmarks

Synthetic suite (50 sites). To obtain a controlled benchmark with automatically recoverable reference sets, we constructed 50 topic-conditioned synthetic websites. We first prepared a fixed inventory of 50 topics using an LLM and stored the resulting topic list as a reusable JSON file. Each topic was then instantiated as a static website from a shared HTML5 template family, yielding lightweight websites with relative intra-site links, page-local styling, and no external JavaScript. This design ensures that evaluation difficulty is driven primarily by site structure and content organization rather than by client-side execution.

To populate each site with realistic multimodal content, we combined LLM assistance with external media retrieval. In particular, topic-specific text and captions were generated with Google Gemini (Google AI for Developers), while image and video assets were validated and, when necessary, replaced with valid URLs retrieved from the Unsplash and Pixabay APIs (Unsplash, n.d.; Pixabay, n.d.), respectively. Because the topic inventory, directory layouts, and asset references are fixed, the reference sets for link discovery and downloadable multimodal extraction can be recovered directly from the generated sites. This enables controlled evaluation under fixed interaction budgets without confounding the analysis with uncontrolled website updates. Additional details, including the synthetic website configuration and reference-set construction procedure, are provided in Appendix C.1.

Real-world websites (online setting). We additionally evaluate on three public websites that stress different failure modes: `berkeley.edu` (dense interactive elements and frequent UI distractions), `imgur.com` (dynamic feed-style pages with heavy client-side rendering), and `loc.gov` (heterogeneous layouts with rich media). All runs are executed online in a headless browser; failures such as non-termination within the time/action budget are treated as unsuccessful runs (reported as "–"). Together, the synthetic and real benchmarks let us separate *controlled set-level evaluation* from *robustness evaluation on live websites*. The synthetic suite primarily serves as a controlled stress test

Benchmark	D	W	P
Synthetic	2	5	10
`berkeley.edu`	2	3	20
`imgur.com`	2	7	50
`loc.gov`	2	3	20

Table 1: Per-benchmark (D, W, P) budgets for internal link discovery (Step 1).

for link discovery and artifact validity, whereas the three real-world websites measure robustness under uncontrolled online variation. Appendix C expands this benchmark construction: Appendix C.1 details the synthetic website configuration and reference-set construction, while Appendix C.3 explains the selection of the three real-world websites.

4.2 Budgets and interaction constraints

For Step 1 link discovery, all methods use the same per-benchmark Step 1 exploration budgets (D, W, P) in Table 1, where D, W, and P denote the maximum BFS depth, the maximum number of retained child links per page, and the maximum number of visited pages, respectively. For page-level extraction, all methods are subject to the same stay-on-page and bounded-scrolling constraints before extracting raw text, image URLs, and video URLs. BaRA additionally applies bounded reflection-guided retries up to the configured retry budget A. Detailed execution settings are reported in Appendix D: Appendix D.1 reports the shared runtime configuration, including the shared backbone model (`gemini-3-flash-preview`), timeouts, and scrolling caps; Appendix D.2 provides the per-benchmark (D, W, P) settings and their rationale; and Appendix D.3 specifies the retry-and-merge rules used under fixed interaction budgets.

4.3 Baselines

We compare against three representative baselines:

- **Pure LLM**: a non-interactive baseline that relies on the backbone model to propose links and/or page items without robust browser-driven control, which often leads to missing dynamic content and hallucinated URLs.
- **SeeAct-Vision** (Zheng et al., 2024): a vision-enabled web agent baseline that operates on live webpages. We use the same backbone model family for fair comparison.

Method	Metric	Dataset			
		Synthetic	Berkeley	Imgur	LOC
Pure LLM	Precision	0.5500	0.0769	0.0400	0.1538
	Recall	0.1834	0.0769	0.1176	0.1538
	Accuracy	0.1190	0.0400	0.0308	0.0833
SeeAct-Vision	Precision	0.4599		0.0000	0.5000
	Recall	0.5299	–	0.0000	0.1538
	Accuracy	0.3526		0.0000	0.1333
Browser-use	Precision	1.0000	0.7692		0.7692
	Recall	0.9220	0.7692	–	0.7692
	Accuracy	0.9220	0.6250		0.6250
BaRA	Precision	1.0000	1.0000	1.0000	1.0000
	Recall	1.0000	1.0000	1.0000	1.0000
	Accuracy	**1.0000**	**1.0000**	**1.0000**	**1.0000**

Table 2: Precision, Recall, and Accuracy for link discovery in Step 1 under fixed budgets (Table 1). "–" indicates non-termination within the budget.

- **Browser-use** (Browser-use contributors, n.d.): an existing browsing agent with the same interaction budgets, but without our bounded BFS traversal and history-based self-reflection logic.

4.4 Evaluation metrics

We evaluate each benchmark using a set-based comparison between the predicted set S_{pred} and the reference set S_{ref}:

$$\text{Precision} = \frac{|S_{\text{pred}} \cap S_{\text{ref}}|}{|S_{\text{pred}}|},$$
$$\text{Recall} = \frac{|S_{\text{pred}} \cap S_{\text{ref}}|}{|S_{\text{ref}}|},$$
$$\text{Accuracy} = \frac{|S_{\text{pred}} \cap S_{\text{ref}}|}{|S_{\text{pred}} \cup S_{\text{ref}}|}.$$

We note that Accuracy denotes the Jaccard index (intersection over union), not standard classification accuracy.

For Step 1, S_{pred} and S_{ref} are sets of internal page URLs. For Steps 2–3, we evaluate text, image, and video separately; text is evaluated using normalized token sets, and image/video items are counted as correct only if they match the reference set and are successfully downloadable. We report both the oracle-link and end-to-end settings for Steps 2–3. Full evaluation details are provided in Appendix A: Appendix A.1 specifies the normalization and download-validity rules, Appendix A.2 defines the BFS-consistent end-to-end setting, Appendix A.3 reports the full Precision/Recall/Accuracy tables for both settings, and Appendix A.4 explains how to interpret a link-discovery Accuracy of 1.0 under this protocol.

Method	Metric	Synthetic			Berkeley			Imgur			LOC		
		Text	Image	Video	Text	Image	Video	Text	Image	Video	Text	Image	Video
Pure LLM	Precision	0.2692	0.0000	0.0000	0.4415	0.0000	0.0000	0.3674	0.0000	0.0000	0.6659	0.0204	0.0000
	Recall	0.1692	0.0000	0.0000	0.2351	0.0000	0.0000	0.1548	0.0000	0.0000	0.2580	0.0065	0.0000
	Accuracy	0.1160	0.0000	0.0000	0.1831	0.0000	0.0000	0.1322	0.0000	0.0000	0.2261	0.0050	0.0000
SeeAct-Vision	Precision	0.9792	0.0000	0.0000	–	–	–	0.0330	0.0000	0.0000	0.9785	0.1667	0.0000
	Recall	0.8651	0.0000	0.0000	–	–	–	0.1364	0.0000	0.0000	0.7376	0.1563	0.0000
	Accuracy	0.8566	0.0000	0.0000	–	–	–	0.0273	0.0000	0.0000	0.7264	0.0877	0.0000
Browser-use	Precision	0.8832	1.0000	0.8148	0.7451	0.9487	1.0000	–	–	–	0.9334	0.9387	0.0000
	Recall	0.8723	0.7297	0.8800	0.6343	0.5506	0.3333	–	–	–	0.7696	0.7897	0.0000
	Accuracy	0.8519	0.7400	0.7400	0.5212	0.5347	0.3333	–	–	–	0.7295	0.7509	0.0000
BaRA	Precision	0.9607	0.9769	1.0000	0.7729	1.0000	1.0000	0.4604	0.4545	0.5429	0.8137	0.8834	1.0000
	Recall	0.9918	1.0000	1.0000	0.8903	0.6726	0.3333	0.6487	0.1974	0.5714	0.9357	0.9960	1.0000
	Accuracy	**0.9570**	**0.9600**	**1.0000**	**0.7058**	**0.6726**	**0.3333**	**0.3808**	**0.1854**	**0.4000**	**0.7706**	**0.8803**	**1.0000**

Table 3: Precision, Recall, and Accuracy for downloadable multimodal extraction (text, image, video) in the **oracle-link setting**. Steps 2–3 are evaluated on the full reference set of target page URLs, so the scores reflect extraction quality independent of Step 1 misses in link discovery.

5 Results and Analysis

We first present quantitative results for link discovery and downloadable multimodal extraction, and then analyze a qualitative case study that illustrates the main failure modes and BaRA's advantages.

5.1 Quantitative Results

We observe three consistent patterns in Tables 2 and 3. First, bounded BFS substantially improves page coverage under fixed budgets. Second, good link coverage does not automatically translate to usable downloadable multimodal extraction: reaching relevant pages still does not guarantee download-valid images or videos. Third, dynamic websites amplify the gap between generic web interaction and robust data collection, so recovery and post-processing become particularly important.

These patterns also explain why we analyze Step 1 and Steps 2–3 separately. Step 1 determines which internal page URLs enter the downstream pipeline. Steps 2–3, then determine whether those visited pages yield download-valid artifacts after constrained per-page extraction and validation. Strong link discovery is therefore necessary but not sufficient: once Step 1 misses a relevant page, downstream extraction never sees that page, and once Step 2 returns weak media outputs, strong traversal alone does not recover the missing artifacts. This separation keeps the quantitative analysis aligned with the pipeline structure.

Link discovery. We report the accuracy for link discovery in Table 2. BaRA achieves the oracle BFS reference set (Accuracy = 1.0000) on all four benchmarks, indicating that bounded BFS traversal reliably enumerates the target set of internal page URLs under the given budgets. This result is particularly meaningful because the strongest runtime baseline, Browser-use, is already competitive on the synthetic benchmark (0.9220), but still falls short on all real websites, dropping to 0.6250 on both `berkeley.edu` and `loc.gov`, and not terminating on `imgur.com`. The Pure LLM baseline performs poorly across the board (0.1190 on Synthetic, below 0.1 on Berkeley/Imgur/LOC), suggesting that direct prompting without robust browser control is insufficient for reliable site-level traversal.

The behavior of the vision-based agent is also informative. SeeAct-Vision reaches 0.3526 on the synthetic benchmark, does not terminate on `berkeley.edu`, fails on `imgur.com`, and reaches 0.1333 on `loc.gov`. In our Berkeley run, the page exposed many interactive elements, causing the agent to get stuck before finishing exploration within the budget. Together, these results suggest that *coverage-oriented control*, rather than stronger grounding, is critical for robust link discovery on real-world websites.

Downloadable multimodal extraction. We report the accuracy for text and for the download-valid image and video items in Table 3. BaRA is consistently stronger in text and, most notably, in *media extraction*, the most fragile part of the pipeline. On the synthetic benchmark, BaRA reaches 0.9570/0.9600/1.0000 on text/image/video, outperforming Browser-use by large margins on image (+0.2200) and video (+0.2600). This result shows that the gains are not limited to navigation: even when the runtime baseline is already reasonably strong, our constrained per-page extraction and recovery steps yield more complete outputs.

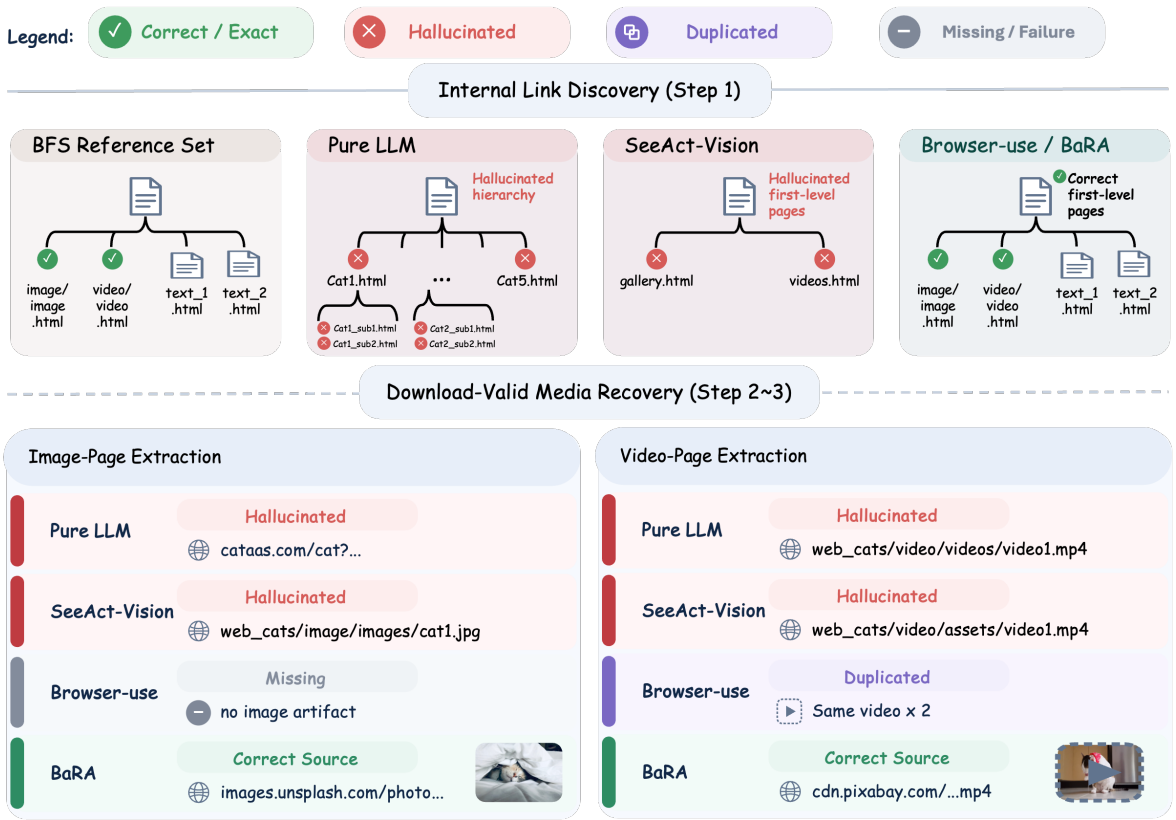

Figure 2: Synthetic cat website case study. **Top: link discovery relative to the BFS reference set.** Pure LLM produces a hallucinated hierarchy, SeeAct-Vision predicts non-existent first-level pages, and both Browser-use and BaRA recover the correct first-level content pages. **Bottom: download-valid media recovery on the image and video pages.** Pure LLM and SeeAct-Vision return hallucinated URLs, Browser-use misses the final image artifact and duplicates one video URL, and BaRA recovers the correct media sources used by the site generator.

A second observation is that *navigation success and artifact success are not the same*. For example, Browser-use obtains relatively strong link discovery on Synthetic (0.9220) and moderate link discovery on `loc.gov` (0.6250), yet its download-valid video Accuracy on `loc.gov` is still 0. Similarly, SeeAct-Vision attains fairly high text Accuracy on Synthetic (0.8566) and `loc.gov` (0.7264), but remains near zero on images and videos. These gaps indicate that reaching the right pages is only the first half of the problem: the system must also preserve page context, expose lazy-loaded content, output URLs in machine-processable form, and return resources that are actually downloadable.

A third observation is that the hardest websites amplify differences between methods. On `berkeley.edu`, BaRA substantially improves over Browser-use on text (0.7058 vs. 0.5212) and image (0.6726 vs. 0.5347), while matching its video score (0.3333). This pattern suggests that once exploration has stabilized, the main remaining challenge in Berkeley is not discovering the site structure but extracting complete multimodal content from cluttered and highly interactive pages. On `imgur.com`, the distinction is even sharper: all baselines fail to retrieve any valid image or video URLs, whereas BaRA still obtains non-trivial scores on text (0.3808), image (0.1854), and video (0.4000). This is the clearest evidence that our recovery and post-processing pipeline helps in a genuinely dynamic setting, rather than only in controlled or static environments, where the underlying page state is easier to preserve.

Finally, `loc.gov` shows BaRA's reliability as well as its ability to recover richer media artifacts than the baselines. BaRA achieves the best score in all three modalities (0.7706 text, 0.8803 image, and 1.0000 video), and is the *only* method with non-zero video Accuracy. Compared with the strongest baseline on images, Browser-use, BaRA improves image Accuracy by more than 13 points (0.8803 vs. 0.7509). This result suggests that our pipeline is particularly effective when websites contain heterogeneous layouts and mixed media types.

Overall takeaway. Taken together, the two tables show that BaRA improves both *where the agent goes* and *what download-valid artifacts it brings back*. The link discovery results isolate the contribution of bounded BFS traversal to predictable coverage, while the multimodal results show that history-based self-reflection, constrained per-page extraction, and download validation are needed to turn that coverage into reliable outputs. The remaining gap on highly dynamic sites such as `imgur.com` also points to a concrete direction for future work: stronger handling of client-side updates, richer page-state monitoring, and more specialized instrumentation for streaming media.

5.2 Qualitative Case Study: Link Discovery and Media Recovery

To illustrate the failure modes behind these aggregate patterns, we present a qualitative case study on a synthetic cat website in Figure 2. The case study summarizes representative outputs for Step 1 (link discovery) and Steps 2–3 multimodal extraction. This example is particularly informative because the site structure is simple and the true media sources are known: the image page is populated with Unsplash image URLs, while the video page is populated with Pixabay-hosted MP4 URLs. The case study, therefore, lets us examine whether an agent follows the actual site structure and recovers the correct media sources used by the site generator, rather than merely retrieving some content.

Link discovery. As shown in the top row of Figure 2, the homepage exposes four first-level content pages: an image page, a video page, and two text pages. Pure LLM instead produces a semantically plausible but structurally unsupported hierarchy, returning URLs such as `cat1.html`, `cat2.html`, and deeper descendants such as `cat1_sub1.html`. SeeAct-Vision also fails to recover the true first-level structure. Although it outputs semantically plausible destinations such as `gallery.html` and `videos.html`, these do not match the actual linked pages, and it additionally omits both text pages. By contrast, both Browser-use and BaRA recover the correct first-level content pages. This comparison suggests that the key distinction is not whether an agent proposes relevant-looking links, but whether the recovered internal page URLs are faithful to the actual reachable site structure.

Download-valid media recovery. The bottom row of Figure 2 shows representative outputs for download-valid image and video recovery on the image and video pages. Pure LLM often produces outputs that are semantically on-topic but poorly grounded in the actual page contents. In this case study, it returns hallucinated URLs rather than the true media sources: on the image page, it produces generic cat-image links, and on the video page, it outputs a video URL that is not actually present on the site. SeeAct-Vision behaves differently from Pure LLM, but its media recovery remains unreliable. In our case study, it predicts hallucinated image and video paths that look plausible for the website, yet do not match the actual URLs embedded in the page. Thus, even when its outputs look more site-specific, they still fail to recover the true downloadable resources.

Browser-use is more faithful to the page structure, but its download-valid media recovery remains brittle. In this case study, it fails to recover any image artifact on the image page, and it extracts the same video twice on the video page. This highlights an important distinction between correct traversal and correct artifact recovery: even when a system reaches the right pages, it may still miss available media or return duplicates.

BaRA is the most consistent with the actual synthetic-site design. On the image page, it recovers the correct Unsplash image source; on the video page, it recovers the correct Pixabay MP4 source. Importantly, it does not replace these assets with hallucinated paths or duplicated files. Taken together, this case study suggests that BaRA's advantage is not simply quantitative. Rather, it more faithfully recovers both the actual internal page URLs and the generator-provided media sources.

6 Conclusion

BaRA makes site-level web data collection more reliable under fixed interaction budgets. We cast collection as set retrieval over internal pages and download-valid artifacts, and show that bounded BFS traversal with history-based self-reflection improves coverage and extraction quality. Across 50 synthetic and three real websites, BaRA outperforms Pure LLM, SeeAct-Vision, and Browser-use, especially on media recovery in dynamic settings. Practical web collection requires explicit coverage control, constrained per-page extraction, and validation of returned artifacts; future work should better handle client-side updates, page-state monitoring, and streaming-media instrumentation.

Limitations

BaRA has two main technical limitations. First, extraction stays on the current page and uses only bounded scrolling, so it misses content revealed through richer interactions such as tabs, accordions, modals, login walls, or multi-step forms. Second, robustness still depends on LLM behavior and browser stability, so results vary across models, providers, and runtime conditions. The framework also carries misuse risks: automated site-level collection could violate website terms, impose unnecessary load, or collect copyrighted or personal content from public pages. We therefore position BaRA as a research prototype and recommend restricting use to public pages, respecting site policies and rate limits, avoiding authenticated or private content, and validating outputs before downstream use. To preserve author anonymity during review, the synthetic benchmark is not included in the current submission and will be released upon acceptance.

References

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault L De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. 2024. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. *Advances in Neural Information Processing Systems*, 37:5996–6051.

Browser-use contributors. n.d. Browser-use: Open-source browser agent runtime for llm-based web interaction. https://github.com/browser-use/browser-use. GitHub repository. Accessed: 2026-03-16.

De Chezelles, Thibault Le Sellier, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F Xu, Siva Reddy, Quentin Cappart, and 1 others. 2024. The browsergym ecosystem for web agent research. *arXiv preprint arXiv:2412.05467*.

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.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, and 1 others. 2024. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*.

Firecrawl. n.d. Firecrawl. https://github.com/firecrawl/firecrawl. GitHub repository. Accessed: 2026-03-16.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. 2023. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arXiv:2305.11854*.

Google AI for Developers. 2025. *Gemini 3 Flash Preview*. Google AI for Developers. Model code: gemini-3-flash-preview. Last updated 2026-02-19 UTC. Accessed: 2026-03-16.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and 1 others. 2024. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*.

Pixabay. n.d. Pixabay developer api. https://pixabay.com/service/about/api/. Accessed: 2026-03-16.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Unsplash. n.d. Unsplash api documentation. https://unsplash.com/documentation. Accessed: 2026-03-16.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. 2025. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2024. Agentoccam: A simple yet strong baseline for llm-based web agents. *arXiv preprint arXiv:2410.13825*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

yt dlp. n.d. yt-dlp. https://github.com/yt-dlp/yt-dlp. GitHub repository. Accessed: 2026-03-16.

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

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2023. Synapse: Trajectory-as-exemplar prompting with memory for computer control. *arXiv preprint arXiv:2306.07863*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Evaluation Details

A.1 Metric details

We use the same Precision, Recall, and Accuracy definitions as in Section 4.4. This appendix specifies the normalization and matching rules used for each modality.

Text matching. For text, both predicted and reference outputs are converted into normalized token sets before scoring. Normalization includes lowercasing and whitespace normalization, and optionally punctuation stripping where applicable. Precision, Recall, and Accuracy are then computed on the resulting token sets.

URL normalization. For link discovery, predicted internal page URLs, image URLs, and video URLs are normalized prior to comparison. This includes fragment removal and minor canonicalization. For images, query parameters and fragments may be stripped to reduce duplicate variants of the same resource.

Download-validity criterion. For image and video items, a prediction is considered correct only if it belongs to the reference set and is successfully downloadable under the same download procedure as the pipeline. Images are downloaded via `requests`; videos are downloaded either via `requests` for direct MP4 links or via `yt_dlp` (yt dlp, n.d.) otherwise.

Empty outputs and failures. If a system returns an empty prediction set, we report Precision, Recall, and Accuracy as 0 for that target set. If a run fails to terminate within the given budget, we mark the corresponding entry as "–".

Evaluation settings. For Steps 2–3, we report both oracle-link and end-to-end settings. In the oracle-link setting, extraction is evaluated on the full reference set of target page URLs. In the end-to-end setting, extraction is evaluated only on the page URLs actually discovered by Step 1.

A.2 BFS-Consistent End-to-End Evaluation

Our goal is not to collect arbitrary existing pages within a site, but to collect data from the pages that should be visited under the given BFS prompt and interaction budget. Accordingly, correctness in the end-to-end setting is defined with respect to an *oracle BFS reference set*, i.e., the set of page URLs that should be reached by a budgeted breadth-first traversal under the specified depth, width, and page constraints.

This distinction matters especially on real websites. A baseline may return URLs that are real and accessible, and it may even collect valid data from those pages. However, if those URLs are not part of the oracle BFS reference set, for example,

Method	Metric	Synthetic			Berkeley			Imgur			LOC		
		Text	Image	Video	Text	Image	Video	Text	Image	Video	Text	Image	Video
Pure LLM	Precision	0.0536	0.0000	0.0000	0.4349	0.0000	0.0000	0.3674	0.0000	0.0000	0.6611	0.0000	0.0000
	Recall	0.0561	0.0000	0.0000	0.2280	0.0000	0.0000	0.1548	0.0000	0.0000	0.2371	0.0000	0.0000
	Accuracy	0.0282	0.0000	0.0000	0.1421	0.0000	0.0000	0.1322	0.0000	0.0000	0.2054	0.0000	0.0000
SeeAct-Vision	Precision	0.8786	0.0000	0.0000	–	–	–	–	–	–	0.9351	0.0000	0.0000
	Recall	0.4351	0.0000	0.0000	–	–	–	–	–	–	0.7163	0.0000	0.0000
	Accuracy	0.4442	0.0000	0.0000	–	–	–	–	–	–	0.6946	0.0000	0.0000
Browser-use	Precision	0.8832	1.0000	0.8148	0.6711	0.4262	1.0000	–	–	–	0.7922	0.9217	0.0000
	Recall	0.8723	0.7297	0.8800	0.6174	0.2321	0.3333	–	–	–	0.6295	0.7937	0.0000
	Accuracy	0.8519	0.7400	0.7400	0.4740	0.1769	0.3333	–	–	–	0.5403	0.7435	0.0000
BaRA	Precision	0.9607	0.9769	1.0000	0.7708	0.9956	1.0000	0.4604	0.4545	0.5429	0.7787	0.8871	1.0000
	Recall	0.9918	1.0000	1.0000	0.9077	0.6756	0.3333	0.6487	0.1974	0.5714	0.8835	0.9964	1.0000
	Accuracy	0.9570	0.9600	1.0000	0.7147	0.6736	0.3333	0.3808	0.1854	0.4000	0.7062	0.8842	1.0000

Table 4: Precision, Recall, and Accuracy for downloadable multimodal extraction (text, image, video) in the **end-to-end setting**. Steps 2–3 operate only on the links actually discovered by Step 1, so traversal misses directly affect downstream extraction performance.

because the method follows a DFS-like path, overcommits to one branch, or otherwise deviates from the frontier expansion implied by the BFS prompt, we do not count the resulting artifacts as correct. In such cases, the extracted content is treated as being collected from the wrong pages relative to the task definition.

Equivalently, the evaluation rewards *policy-consistent collection* rather than arbitrary within-site retrieval. If a method misses a page that should have been visited under the BFS prompt, the artifacts on that page contribute to false negatives. If it instead collects data from a different existing page that lies outside the oracle BFS reference set, those artifacts contribute to false positives. Thus, even valid content from real URLs may be scored as incorrect when it is gathered from the wrong part of the site.

This evaluation choice is deliberate. Without it, a method could receive credit for collecting useful data while still violating the task's core requirement: breadth-first coverage within a fixed interaction budget. Our end-to-end results, therefore, measure whether extraction is performed on the *right pages* under the intended traversal policy, as well as the resulting extraction quality.

A.3 Full Precision/Recall/Accuracy tables

Reporting convention. In Tables 2, 3, and 4, the Synthetic column reports the mean metric value over the 50 synthetic websites. The Berkeley, Imgur, and LOC columns report single-run online results for each method.

For Step 1 link discovery, we report the full Precision/Recall/Accuracy results in Table 2. For downloadable multimodal extraction in Steps 2–3, we use two complementary evaluation settings. Under the **oracle-link setting**, we evaluate extraction and downloading against the full reference set of target page URLs independently of Step 1 traversal outcomes, and we report those results in Table 3. Under the **end-to-end setting**, we measure downstream performance using only the page URLs that Step 1 actually discovers, and we report those results in Table 4. In the Synthetic setting, we predefine the reference set of target page URLs. For real-world websites, we evaluate against a fixed post hoc audit reference constructed independently of the compared methods using the same page-audit procedure and normalization rules.

A.4 On Perfect Link Discovery Accuracy

The link discovery accuracy of 1.0 for BaRA should be interpreted within the evaluation protocol of this work. This result does not mean that the benchmark is easy. Instead, it shows that BaRA follows the bounded BFS traversal policy of Step 1 exactly under the same exploration budget, as reported in Table 1.

The goal of Step 1 is not to retrieve every internal page in a website. Step 1 evaluates whether a method reproduces the bounded BFS policy under the same budget constraints (D, W, P). Therefore, the reference set for link discovery is not the set of all internal pages in the site. The reference set is the set of pages reached by an oracle BFS traversal under the same exploration budget.

In the synthetic benchmark, this reference set is built from websites with static HTML structures and fixed link topology. Under this setting, the site graph is fully inspectable. Given the same depth, width, and page budget, the BFS traversal

result is deterministic. A method that follows the bounded BFS policy exactly therefore recovers the full reference set. In this sense, an accuracy of 1.0 reflects traversal policy fidelity rather than low benchmark difficulty.

This setting is not trivial. Pure LLM, SeeAct-Vision, and Browser-use do not recover the full reference set under the same evaluation protocol and the same exploration budget, as shown in Table 2. This result shows that the main challenge does not lie in the existence of the reference set itself. Rather, the main challenge lies in whether a prompt-driven web agent maintains BFS-style exploration without hallucinated links, premature commitment to a single branch, or incomplete traversal.

B Prompts

B.1 BFS Prompt

The Step 1 prompt was designed to implement the bounded BFS policy at the prompt level by restricting the agent to collecting links only within the same domain as the start URL and its subdomains. This restriction prevents drift toward third-party pages that are not aligned with the data collection objective. In addition, the prompt includes normalization and exclusion rules, such as fragment removal, deduplication of minor URL variants, and filtering of policy pages, login pages, and non-HTML downloadable files, in order to improve the relevance and quality of the collected set of internal page URLs.

```
You are a precise but lightweight BFS
↪  crawler that collects only internal
↪  links.

[Start URL] {first_url}
[Limits] max_depth = {max_depth},
↪  max_width = {max_width}, max_pages =
↪  {max_pages}, output_path =
↪  links_bfs.json

[RULES]
1) Same-site only: links within the
↪  registrable domain of Start URL and
↪  its subdomains.
2) BFS: use a FIFO queue of (url, depth).
↪  Init with (Start URL, 0).
For each dequeued page:
    a) Visit it and extract <a href> in
    ↪  DOM order
    b) Normalize & filter (see 3,4)
    c) Keep first <= max_width unique
    ↪  children for this parent
    d) Enqueue (child, depth+1) if depth+1
    ↪  <= max_depth
Stop when visited pages >= max_pages or
↪  queue is empty.
3) Normalization:
- Remove fragments (#. . . )
- Keep query keys only: [page,p,pg,offset
↪  ,start,sort,category,board]
- http/https and trailing slashes
↪  considered equal
- Deduplicate by (scheme, host, path,
↪  kept_query)
4) Exclude:
- Paths containing: [/privacy, /terms,
↪  /login, /signup]
- File extensions:
↪  [.pdf,.zip,.rar,.7z,.apk,.dmg,.exe]

[OUTPUT]
Return ONLY this JSON and also save to
↪  [links_bfs.json]:
{{
"start_url": "[https://example.com/]",
"max_depth": {max_depth},
"max_width": {max_width},
"max_pages": {max_pages},
"visited_order": [...],
"by_depth": {{"0": [start_url], "1":[...],
↪  ...}}
}}
```

B.2 Failure-Analysis Prompt

This prompt was designed to operationalize the history-based self-reflection mechanism described in Section 3.3. Since BaRA recovers from unsuccessful attempts by revising the next instruction rather than terminating immediately, the prompt explicitly provides the original task together with the accumulated long-term memory and asks the model to identify the failure type, the specific failure point, and concrete prompt-level improvements. The prompt also requires an improved version of the original task so that the reflection step produces an actionable next-step instruction, rather than only a post-hoc explanation of the error. Finally, the response follows a fixed JSON schema, and the structured retry pipeline uses the diagnosis and revised prompt directly.

```
You are an expert in analyzing the causes
↪  of web scraping task failures and
↪  suggesting improvements.

[Original Task]
{original_task}

[Long-term Memory so far (Attempt
↪  {attempt_number})]
{json.dumps(long_term_memories,
↪  ensure_ascii=False, indent=2)}

[Analysis Request]
Please analyze why the above task failed
↪  and provide the following:

1. Failure Analysis:
```

```
   - What type of failure it is (e.g.,
   ↪  insufficient data, access
   ↪  permissions, technical issues,
   ↪  etc.)
   - The specific point of failure
   - Why this failure occurred

2. Improvement Suggestions:
   - Specific instructions that should be
   ↪  added to the prompt
   - Parts that should be clarified
   - Missing considerations that should be
   ↪  included

3. Improved Prompt:
   - A new version of the original task
   ↪  prompt with improvements
   - Concrete steps that address the
   ↪  identified failure causes

Respond in JSON format:
{{
    "failure_analysis": "Detailed analysis
    ↪  of the failure",
    "failure_type": "Type of failure",
    "improvements": ["Improvement 1",
    ↪  "Improvement 2"],
    "improved_prompt": "Improved task
    ↪  prompt"
}}
```

B.3 Success-Evaluation Prompt

This prompt was designed as a lightweight and intentionally generous success evaluator. In BaRA, an attempt is considered useful whenever it yields a usable signal, such as a small number of links, brief but relevant text, or image/video URLs, because the goal is robust collection under fixed interaction budgets rather than all-or-nothing task completion. For this reason, the prompt defines success broadly and marks failure only when no useful data is extracted or the output is clearly irrelevant to the task. Providing both the original task and the final extracted content allows the evaluator to judge whether the produced data is relevant to the intended collection target, as well as whether any data was produced at all. The JSON output further separates the binary decision from its explanation and a short summary of the extracted signal for downstream control and analysis.

```
You are an expert in evaluating the
↪  success of web scraping tasks.

[Original Task]
{original_task}

[Last Extracted Content (Final Result)]
{json.dumps(evaluation_data,
↪  ensure_ascii=False, indent=2)}

[Evaluation Request]
Please determine whether the above task
↪  was successful or not.

Success Criteria (Task is SUCCESSFUL if
↪  ANY of the following is true):
1. Links were successfully extracted from
↪  the page (even if only a few)
2. Text content was extracted and contains
↪  relevant information
3. Images or videos were found and
↪  extracted
4. The scraping process completed without
↪  major errors
5. The extracted content is related to the
↪  original task requirements

IMPORTANT: Be GENEROUS in your evaluation.
↪  If the task extracted ANY useful data
↪  (links, text, images, videos),
↪  consider it a SUCCESS. Only mark as
↪  FAILURE if:
- No data was extracted at all
- The extraction process completely failed
- The extracted content is completely
↪  irrelevant to the task

Specific Evaluation Points:
- Check if links array contains valid URLs
↪  (even if just a few)
- Check if text content was extracted
↪  (even if brief)
- Check if images or videos were found
- Consider the task successful if it
↪  gathered useful information, even if
↪  not perfect

Respond in JSON format:
{{
    "is_successful": true/false,
    "reason": "Detailed reason for
    ↪  success/failure based on what was
    ↪  actually extracted",
    "extracted_data_summary": "Summary of
    ↪  what was actually extracted and
    ↪  why it should be considered
    ↪  success/failure"
}}
```

B.4 Single-Page Extraction Prompt

This appendix reports the Step 2 single-page extraction prompt used in the constrained per-page extraction protocol. The agent must remain on the target URL, use bounded scrolling, and return raw text together with candidate image and video URLs in a schema-constrained format. These candidate URLs are then normalized and passed to Step 3 download validation in the downstream pipeline. This design is motivated by the fact that the stability of this stage depends more on the length and parseability of the textual output returned by the agent than on the underlying media files themselves. Consequently, instead of directly downloading im-

ages and videos, the prompt requires the agent to list image URLs and video URLs. The extracted URLs are then separately validated, normalized, and downloaded by dedicated functions in subsequent stages. This separation of responsibilities is consistently maintained throughout the downstream implementation.

```
Step 1: Go to {sub_url} and STAY on this
↪  specific page.
Step 2: Restricted Extraction (Single Page
↪  Only)
- DO NOT click any links, DO NOT navigate
↪  to other URLs.
- Mandatory: Scroll to the bottom
↪  repeatedly until no new content loads.
- Safety cap: stop scrolling if you have
↪  performed {MAX_SCROLL_ACTIONS} scroll
↪  actions, or spent {MAX_SCROLL_SECONDS}
↪  seconds on scrolling, or if page
↪  height does not increase for
↪  {NO_GROWTH_CYCLES} consecutive scroll
↪  cycles. After hitting any cap, proceed
↪  to extraction immediately.
Step 3: Comprehensive Data Extraction
- Extract EVERY piece of raw text, image
↪  URLs, and video URLs.
- All available image URLs (ending with
↪  .jpg, .png, .gif, .jpeg) that appear
↪  on that page.
- All available video URLs (ending with
↪  .mp4, .mov, .avi, .wmv, .flv, .mkv,
↪  .webm, .m4v, .m3u8, .ts) that appear
↪  on that page.
Step 4: Strict Output Format for
↪  Post-Processing
- Provide the data using the following
↪  specific delimiters.
- Do NOT summarize or use '...' to
↪  truncate text. List everything found.
- If a category is empty, write 'None'
↪  between the tags.

[Page URL]

- Text(s): (Extract all text blocks here.
↪  Each paragraph or block must be on a
↪  new line. No bullet points or
↪  decorative symbols)

- Image(s): (List all full Image URLs here,
↪  one per line)

- Video(s): (List all full Video URLs here,
↪  one per line)
```

C Benchmark Construction Details

C.1 Synthetic Website Configuration

We built the synthetic corpus by explicitly separating *site topology* from *topic instantiation*. For each of the 50 topics in the stored topic inventory, the generator first selected a layout specification from a small family of HTML5 site templates and then filled that structure with topic-conditioned titles, short descriptions, section headers, navigation labels, and captions. Navigational pages use relative internal links, whereas downloadable artifacts remain absolute remote URLs. This separation was useful for benchmark design because it preserves a fixed, inspectable link graph for evaluation while still allowing the site's visible content to vary substantially across topics.

Layout family and depth control. The template family was designed to vary the difficulty of the traversal without altering the basic multimodal objective. In the shallowest configuration, all content for a topic is presented in a single `index.html`. A second configuration uses a small multi-page site in which the root page links to dedicated image and video pages together with one or more text pages (e.g., `image/image.html`, `video/video.html`, and `text_k.html`). A deeper variant inserts a portal-style entry page at the root and nests the substantive content site under a secondary directory such as `section_level_2/`. These variants preserve comparable topical coverage while systematically varying the number of navigation decisions required before the agent reaches the pages containing the relevant evidence. As a result, the synthetic suite stresses both direct page-level extraction and more structured multi-hop link discovery under the same evaluation protocol.

Content population and media handling. Topic-specific copy was produced with Google Gemini, which generated short natural-language passages and captions intended to be informative but not excessively repetitive across sites. We intentionally used the model for semantic framing rather than as the final authority on downloadable media URLs. Candidate image and video assets were therefore validated before inclusion, and any broken, unstable, or otherwise unsuitable references were replaced with verified assets retrieved from Unsplash and Pixabay, respectively. In other words, the textual layer is model-assisted, whereas the media layer is grounded by explicit URL validation. We also allowed modest variation in the number of media items and text subpages across topics so that agents could not rely on a single fixed cardinality pattern when collecting artifacts.

Reference recovery and benchmark scope. The benchmark reference sets are derived directly from

the generated static sites. For link discovery, we derive the reference set of internal page URLs from the released directory layout. For downloadable multimodal extraction, we collect the text-bearing pages along with the validated image and video URLs embedded in them, and apply the same normalization rules used during scoring. This design makes the benchmark fully inspectable: once the corpus is generated, repeated runs operate over the same site graph and the same download-valid inventory. We intentionally excluded authentication, form submission, and heavy client-side rendering from this synthetic suite. Those behaviors are covered separately by the real-world benchmark; here, we use a controlled environment that isolates link discovery and page-level extraction errors from nondeterministic website updates.

C.2 Reference-Set Construction for the Synthetic Benchmark

The synthetic reference sets are derived from a prepared static website corpus and compiled into a single reference table. For each site, we enumerate the HTML pages under the site directory and map them to their corresponding hosted page URLs. For link discovery evaluation, the Step 1 reference set is defined as the subset of these URLs reached by the budgeted oracle BFS policy under the same (D, W, P) constraints used during evaluation. We then parse the same HTML pages to collect the image and video artifacts referenced by the site and aggregate them at the site level. The resulting table stores the site identifier, structural depth, counts of pages, images, and videos, and the corresponding URL inventories.

Because the synthetic corpus is static, this construction is deterministic and does not require manual annotation. During evaluation, system outputs are compared against this reference table, with the normalization and download-validity rules described in Appendix A.1 applied at scoring time. Upon acceptance, we will release the synthetic benchmark, including the website corpus and `web_stats.csv`, allowing the benchmark to be fully inspected and the reference sets to be reproduced without access to the internal data-generation pipeline.

C.3 Real-World Website Selection

To address the static nature of data collection in the synthetic setting, we additionally constructed a real-world evaluation set using three publicly accessible websites: Imgur (imgur.com), Library of Congress (loc.gov), and UC Berkeley (berkeley.edu). Although the synthetic environment allows controlled experimentation, it does not fully capture the complexity of live web environments, where page layouts change dynamically, content structures are heterogeneous, and interaction patterns vary substantially across websites. We therefore selected these three websites to evaluate our agent's robustness in collecting text, image, and video content under realistic web conditions.

Our selection was guided by three considerations. First, we aimed to evaluate the agent in live, open-world web environments rather than in simulated or template-driven settings. This is important because real-world deployment requires robustness to continuously changing interfaces and diverse page organizations. Second, we chose websites that provide complementary multimodal content. Imgur represents a dynamic media-centric platform with image- and video-oriented content and highly interactive pages. Loc.gov provides a wide range of publicly accessible archival materials, including text documents, images, audio, and video collections. Berkeley.edu provides text-rich informational pages together with image-heavy news content and video-linked materials, offering a different institutional website structure. Third, we intentionally included websites with different interaction and navigation characteristics to test whether our agent maintains stable collection performance across substantially different real-world website types.

In general, we selected these websites not to optimize for a particular benchmark difficulty but to test whether the proposed agent reliably collects multimodal content from heterogeneous, live, and practically relevant web environments.

C.4 Real-World Reference-Set Construction

For real-world websites, it is difficult to fully predefine ground-truth reference sets in advance because the target pages are live and time-varying. We therefore construct a fixed post hoc audit reference for evaluation, independent of the compared methods. Concretely, we apply the same page-audit procedure to a common audit URL set for each website and record the observable image URLs, video URLs, internal links, and full page text in `page_i.json`. Final metrics for all methods are then computed against this shared audit reference, using the same normalization and download-

Parameter	Value
Headless mode	True
Max actions per step	1
LLM provider	Google Gemini
Base model name	gemini-3-flash-preview
Max (retry) attempts per stage	2
Random seed	42
Step 1 temperature	0.6
Step 1 top-p	1.0
Step 1 LLM timeout	240 s
Step 1 step timeout	300 s
Step 1 total timeout	1600 s
Step 2 temperature	0.6
Step 2 top-p	0.7
Step 2 LLM timeout	180 s
Step 2 step timeout	240 s
Step 2 total timeout	500 s
Step 2 supplemental retry total timeout	1000 s
Step 3 batch size	50
Firecrawl (Firecrawl, n.d.) integration	Enabled only when an API key is provided
API Keys	Enabled only when an API key is provided

Table 5: Runtime configuration used across experiments. Per-benchmark settings (D, W, P) are reported separately.

validity rules as in the synthetic setting. These audit annotations are used only for post hoc evaluation and are not exposed to any method during browsing, extraction, or verification.

D Experimental Settings

D.1 Runtime Configuration

All experiments were conducted in a Browser-use-based headless Chromium environment. The shared runtime configuration is summarized in Table 5. In particular, we used `gemini-3-flash-preview` for Step 2 page-level extraction under the Google provider, motivated by the practical importance of throughput and response latency in repeated browser-agent execution.

D.2 Per-benchmark Budget Settings

For internal link discovery, we used the same (D, W, P) budgets reported in Table 1 of the main paper, where D denotes the maximum BFS depth, W the maximum number of retained child links per page, and P the maximum number of visited pages. Specifically, we set $(D, W, P) = (2, 5, 10)$ for the Synthetic benchmark, $(2, 3, 20)$ for `berkeley.edu`, $(2, 7, 50)$ for `imgur.com`, and $(2, 3, 20)$ for `loc.gov`. For each benchmark, we manually chose D, W, and P as the smallest values that still allowed Step 1 to reach image-bearing and video-bearing pages under the benchmark's BFS regime; the final selected values are reported in Table 1. All other runtime parameters were fixed across benchmarks as summarized in Table 5.

These allocations were chosen to reflect differences in website structure and interaction characteristics. The Synthetic benchmark consists of lightweight static sites drawn from a small family of depth-controlled layouts, with recoverable reference sets and limited within-site branching, so a relatively small page budget is sufficient for representative coverage. In contrast, the real-world websites require larger budgets because they exhibit heterogeneous layouts and uncontrolled online variation. We used a moderate budget for `berkeley.edu` and `loc.gov`, whose page organizations are relatively structured, while assigning a larger width and page budget to `imgur.com` due to its feed-style interface, aggressive lazy loading, and progressively revealed content. This budget design was intended to keep exploration bounded while still allowing the agent to recover a representative set of internal page URLs under each website's interaction regime.

D.3 Retry-and-Merge Rules

We summarize the retry-and-merge rules used in BaRA under fixed interaction budgets in Table 6. In line with Section 3.3, BaRA uses history-based self-reflection to revise the next instruction based on the agent's execution history and the partial outputs observed so far. At runtime, this mechanism is realized as bounded reflection-guided retries. Each stage is first executed with the original

Stage	Rule summary
Step 1 (link discovery)	Retry when the traversal record is invalid, timed out, malformed, or otherwise unusable. The final output is replacement-based: keep one coherent traversal output from a single bounded BFS traversal and do not merge discovered page URLs across attempts.
Step 2 (page-level extraction)	Retry when rendering is incomplete or the extraction is malformed, empty, or partial. The final output is merge-based for the same page: concatenate text blocks after trimming empty lines and removing exact duplicates, then normalize and deduplicate image URLs and video URLs before download validation.
Step 3 (download validation)	Applied after Step 2 aggregation. This stage retains download-valid artifacts in the final outputs.

Table 6: Stage-wise summary of the retry-and-merge rules used in BaRA under fixed interaction budgets.

instruction, and a retry is triggered only when the attempt does not yield an acceptable output under the stage-specific criterion.

For Step 1 (link discovery), retries are replacement-based rather than union-based. If an attempt fails to produce a valid traversal record, the system re-runs the bounded BFS traversal with the revised instruction and retains one coherent final traversal output. We do not merge discovered page URLs across multiple Step 1 attempts. This choice is important because naïvely taking the union of partially successful traversals would break the bounded BFS traversal semantics being evaluated and could artificially inflate link coverage with page URLs that were not recovered within one budget-consistent run.

For Step 2 (page-level extraction), which instantiates the constrained per-page extraction protocol, retries are merge-based at the level of extracted content for the same page. When multiple attempts are available, their outputs are aggregated modality-wise before downstream validation. Text blocks are concatenated in attempt order after trimming empty lines and removing exact duplicates. Image URLs and video URLs are normalized under the same canonicalization rules used during scoring and then deduplicated prior to download validation. The resulting candidate set is passed to download validation, which removes download-invalid artifacts and retains only download-valid artifacts in the final outputs. This design is intentionally recall-oriented: complementary attempts may expose additional lazy-loaded content or recover items omitted by an earlier run.

Malformed outputs are handled conservatively throughout the retry pipeline. Outputs that fail the required schema, contain no usable signal, or are clearly irrelevant to the target page are treated as unsuccessful attempts and trigger repair when the remaining budget allows. At the same time, partially useful extractions are not discarded unnecessarily: if an attempt returns plausible text or media evidence for the correct page, those candidates may still contribute to the merged set, with final correctness determined only after normalization and download validation.

E Additional Real-World Website Case Studies

To complement the aggregate results in Table 3, we provide three additional real-world website case studies here: `https://www.berkeley.edu/`, `https://imgur.com/`, and `https://www.loc.gov/`. These examples explain why similar metric trends arise from qualitatively different failure modes.

Berkeley. Berkeley is a cluttered, interaction-heavy university website with multiple subdomains and mixed media. Pure LLM again tends to generate Berkeley-like media addresses that look plausible but are not grounded in the actual validated artifacts, which is consistent with its low Berkeley scores in Table 3 (0.1831 / 0.0000 / 0.0000 for text/image/video). SeeAct-Vision does not terminate on this site, suggesting that the large number of interactive elements likely traps the vision-based agent before it produces a stable extraction. Browser-use is already reasonably strong on Berkeley, reaching 0.5212 / 0.5347 / 0.3333. BaRA nevertheless improves text and image extraction to 0.7058 and 0.6726 while matching the same video score of 0.3333. Thus, Berkeley is not a case where only BaRA works; rather, it is a case where multiple browser-based methods recover useful content, but BaRA is more complete on text and images under the same interaction budget.

Imgur. Imgur exhibits a qualitatively different difficulty profile. Its challenge lies in aggressive

lazy loading in a large feed-style interface, as well as in noisy interactions. Pure LLM behaves similarly to the Berkeley case, producing semantically plausible but weakly grounded outputs. SeeAct-Vision exhibits some exploratory behavior but fails to produce a stable traversal artifact and ultimately achieves only weak text performance, with no image/video recovery. Browser-use collapses entirely to zero on all three modalities. By contrast, BaRA's logs show that the system recovers from an accidental detour, remains on the target page, and continues page-level extraction under bounded scrolling while additional content is progressively revealed as the gallery expands. This is consistent with Table 3, where BaRA is the only method that achieves non-zero image and video Accuracy on Imgur (0.1854 and 0.4000), together with the strongest text score (0.3808). The Imgur case, therefore, suggests that reflection and recovery are particularly useful when content is progressively revealed during scrolling rather than fully exposed in the initial DOM.

LOC. On `https://www.loc.gov/`, all methods produce usable runs, making it a cleaner comparison point than Berkeley or Imgur. Even so, Table 3 still shows a clear difference in artifact quality. Pure LLM remains weak overall (0.2261 / 0.0050 / 0.0000), while SeeAct-Vision is already fairly strong on text (0.7264) but much weaker on image extraction (0.0877) and still fails on video. Browser-use is competitive on both text and images (0.7295/0.7509), but still recovers no valid video artifacts. BaRA achieves the best result in all three modalities (0.7706 / 0.8803 / 1.0000) and is the only method with non-zero video performance. In this sense, LOC complements Imgur: Imgur highlights recovery under dynamic loading, whereas LOC highlights completeness on a comparatively structured public website with mixed media.

F End-to-End Case Study on the Synthetic Cat Website

We provide an additional case study for the *end-to-end* setting here. Unlike the oracle-link setting, Steps 2–3 in this setting operate only on the links actually discovered by Step 1. As a result, traversal errors propagate directly into downstream extraction quality.

Pure LLM. Pure LLM illustrates the most severe form of end-to-end error propagation. Its downstream extraction is not necessarily sparse: in fact, it collects substantial amounts of cat-themed text, images, and videos. However, this content is gathered from a hallucinated traversal structure, including pages such as `cat1.html`, `cat1_sub1.html`, and `cat1_sub2.html`, rather than from the true target pages. One of the traversed pages even fails with a 503 error. This explains why the end-to-end scores in Table 4 remain poor despite apparently rich extracted outputs: The problem lies in the extraction of the wrong pages as well as in the extraction quality.

SeeAct-Vision. SeeAct-Vision fails earlier in the pipeline. As already illustrated in Figure 2, its `links_bfs.json` overlaps with the oracle BFS traversal only partially: among the three discovered URLs, only one is correct. Consequently, its end-to-end behavior is dominated by Step 1 mismatch. Even if the extractor preserves some page-relevant text, it has little chance to recover the target image and video resources when the traversal itself does not reach the correct media pages.

Browser-use. Browser-use shows a different pattern. Its Step 1 traversal is correct and recovers the same five-page structure as the oracle BFS traversal. Accordingly, its end-to-end degradation is mainly driven by Step 2 extraction rather than by link discovery. In particular, the image page extraction repeatedly fails to produce a stable final artifact, whereas the video page succeeds but results in duplicate downloads. The final media aggregate is heavily skewed toward videos, with no images and seven saved videos. This case highlights that correct traversal alone is not sufficient for strong end-to-end performance: once the pipeline reaches the right pages, robustness still depends on whether extraction returns the right artifacts without omission or duplication.

BaRA. BaRA also recovers the correct five-page traversal, so its end-to-end behavior is likewise governed primarily by extraction quality. Compared with Browser-use, however, BaRA is much more reliable on the media-bearing pages themselves. On the image page, it recovers the five Unsplash image URLs used by the synthetic generator, and on the video page, it recovers the three Pixabay MP4 URLs. This gives BaRA a strong advantage in end-to-end image and video collection. At the same time, the logs reveal a remaining source of error: on text-focused pages, retry-and-merge occa-

sionally introduces hallucinated media URLs that subsequently fail to download, resulting in 404 errors. Thus, the synthetic end-to-end case study suggests that BaRA's remaining gap lies less in link discovery and more in residual false-positive media predictions during page-level extraction on pages that should remain mostly textual.

Takeaway. Overall, the synthetic end-to-end setting separates the methods into three distinct regimes. Pure LLM mainly fails because it extracts from the wrong pages; SeeAct-Vision mainly fails because its partial traversal prevents downstream recovery; Browser-use reaches the correct pages but is brittle at downloadable multimodal extraction; and BaRA benefits from both correct traversal and stronger recovery on the true media pages, while still leaving room for improvement in suppressing false-positive media on text-centric pages.

G Additional End-to-End Real-World Website Case Studies

We provide three additional real-world website case studies for the *end-to-end* setting here. Unlike the oracle-link setting, Steps 2–3 are executed only on the links actually discovered by Step 1, so traversal errors directly propagate to downstream extraction performance. As a result, the same website exposes a different bottleneck in the end-to-end regime than in the oracle-link analysis.

Berkeley. Berkeley remains an interaction-heavy and operationally unstable site in the end-to-end setting. Pure LLM is weak because of unstable downstream processing as well as low-quality extraction: in our run, many Berkeley pages in the discovered set returned repeated 503 errors before a usable page-level result could be produced. SeeAct-Vision has no Berkeley end-to-end entry because the agent gets stuck before completing traversal. Browser-use remains non-trivial on Berkeley, but its end-to-end performance is clearly lower than its oracle-link performance. This is consistent with the logs: the crawl is interrupted before all planned pages are processed, and one of the extracted pages (the Berkeley map page) is explicitly flagged as truncated and incomplete. BaRA performs best on Berkeley in the end-to-end regime, improving both text and image extraction while matching Browser-use video score. In other words, Berkeley is a case where end-to-end performance is limited by a combination of partial traversal, repeated runtime failures, and incomplete page-level extraction, rather than by media validation alone.

Imgur. Imgur is the clearest example of how end-to-end evaluation amplifies dynamic-page failures. Pure LLM retains only a weak text signal and still fails completely on media, indicating that topic-consistent outputs are not enough once traversal and download validity are considered. SeeAct-Vision also fails to produce a stable crawl artifact for Imgur, so there is effectively no reliable downstream media recovery. Browser-use collapses entirely to zero on text, image, and video. Its logs show repeated DOM/state failures and an empty extraction on the first gallery page, indicating the pipeline never stabilizes enough to produce download-valid artifacts. By contrast, BaRA is the only method that maintains non-zero performance on all three modalities. The logs show why: after recovering from an initial detour, it stays on the target page, continues bounded scrolling, and keeps extracting as the gallery progressively reveals more content, with the page expanding from roughly 32k pixels to over 40k pixels in height. Thus, the Imgur case suggests that end-to-end robustness on dynamic galleries depends on the ability to recover, continue past incomplete states, and extract from progressively revealed content during scrolling, as well as on initial navigation.

LOC. The Library of Congress presents a different end-to-end profile. Here, all methods produce at least some usable outputs, so the main question is not whether the pipeline runs at all, but how much valid multimodal content survives the full traversal and extraction chain. Pure LLM remains weak overall and recovers no valid video content. SeeAct-Vision achieves a relatively strong text score, but its zero image and video performance in the end-to-end setting indicates that limited link coverage and media extraction still constrain the final output. Browser-use performs reasonably well on text and image, but continues to miss video entirely. This matches the logs, where image extraction succeeds on LOC pages while the extracted page summaries explicitly report no valid video URLs. BaRA again performs best in the end-to-end regime and is the only method with non-zero video performance. Its logs show successful recovery of the homepage and a directly downloadable MP4, along with a broad image collection. LOC, therefore, complements Imgur: whereas Imgur stresses recovery un-

der highly dynamic loading, LOC highlights the importance of completeness once the site structure itself is comparatively stable.

End-to-end takeaway. Across Berkeley, Imgur, and LOC, the end-to-end setting exposes three different failure regimes. Berkeley is mainly limited by runtime instability and incomplete extraction on partially covered crawls; Imgur is dominated by dynamic loading and the need for explicit recovery after incomplete states; and LOC is comparatively stable, so the remaining gap is largely one of multimodal completeness. These cases support the broader point of the paper: strong per-page extraction in isolation is not sufficient for robust web data collection unless the system also preserves coverage and artifact validity throughout the full pipeline.