Does Format Matter? Agent Task Performance Across Web Representations

David Hurley
Plasmate Labs

March 2026

Abstract

Large language model agents frequently operate over raw HTML, despite the fact that HTML is optimized for browser rendering rather than machine comprehension. We introduce WebTaskBench, a benchmark of 100 agent tasks spanning five categories and 50 real-world websites, and evaluate how three web representations affect system cost and speed: raw HTML, cleaned markdown, and a structured Semantic Object Model (SOM). Across two models (GPT-4o and Claude Sonnet 4) and three runs per task, SOM reduces average input tokens by 4.0x versus raw HTML (33,181 to 8,301 tokens per task). While markdown is smaller than SOM (7.3x reduction versus HTML), SOM achieves lower latency on both models and is substantially faster on Claude (8.5s average for SOM vs 25.2s for markdown). These results suggest that structured representations can provide efficiency gains beyond simple compression, and motivate further evaluation of accuracy and hallucination outcomes.

1 Introduction

Web browsing agents are now used for research, customer support, monitoring, and automation. Yet most agent pipelines still provide raw HTML to language models. HTML is optimized for pixels, not meaning. It contains navigation chrome, scripts, styles, and repeated boilerplate that (1) inflates token cost, (2) reduces effective context capacity, and (3) forces the model to infer structure implicitly.

A common alternative is to convert pages into markdown or plain text using readability-style extraction. This often reduces tokens substantially, but it can also flatten structure and remove information that is critical for navigation tasks (for example, which elements are buttons, which text labels an input, or what belongs to which region).

This paper asks a simple question: does representation format matter for agent workloads beyond compression? We introduce WebTaskBench, a benchmark that measures task-oriented agent behavior across web representations. In this revision we report completed measurements for cost (tokens) and speed (latency). WebTaskBench is designed to also measure task accuracy and hallucination rates. We specify the scoring rubrics and evaluation framework required to compute those metrics and will report the resulting outcomes in subsequent revisions.

Contributions. (1) WebTaskBench, a task-based benchmark for web representations; (2) a reproducible execution harness using cached corpus snapshots; (3) empirical results for token consumption and latency across GPT-4o and Claude Sonnet 4; (4) category-level analysis showing the largest compression gains for navigation and adversarial pages.

2 Related work

Web agent benchmarks. Prior benchmarks evaluate web navigation and task completion, often using live websites or interactive browser environments [1, 2, 3]. These settings test end-to-end agent behavior, but typically do not isolate the impact of the page representation format.

Grounding and hallucination. Web agents must ground answers in source content and avoid unsupported claims. Work on browser-assisted answering and agent prompting highlights the difficulty of reliable grounding [4, 5].

Representations for LLM consumption. A growing ecosystem of tools provides markdown-like or text representations of web pages to language models. These methods reduce token cost but often discard semantic typing and region structure that may be useful for navigation and UI understanding.

3 Background: the Semantic Object Model (SOM)

SOM is a structured JSON representation produced by Plasmate. It encodes page regions (for example, header, main, nav, footer), semantic element types (text, link, button, input), and attributes such as labels and destinations. SOM aims to preserve meaning and structure while removing rendering-specific details.

Listing 1 shows a schematic example of the kind of information preserved in SOM.

Listing 1: Illustrative SOM fragment (schematic).

```
{
  "regions": [
    {
      "role": "main",
      "elements": [
        {"type": "heading", "text": "Pricing"},
        {"type": "button", "text": "Start free trial"},
        {"type": "link", "text": "Docs", "href": "/docs"}
      ]
    }
  ]
}
```

In contrast, markdown extraction primarily preserves visible text with limited semantic typing. Raw HTML preserves the full DOM, but requires the model to infer what is salient.

4 WebTaskBench

4.1 Benchmark definition

WebTaskBench contains 50 URLs spanning news, ecommerce, reference, government, social, SaaS, and adversarial pages (Appendix A). The benchmark defines 100 tasks across five categories: extraction, comparison, navigation, summarization, and adversarial noise resistance.

Category	Tasks
Extraction	20
Comparison	20
Navigation	20
Summarization	20
Adversarial	20

Table 1: WebTaskBench task categories (100 tasks total).

4.2 Representations and caching

For each URL we cache three representations: raw HTML, a cleaned markdown-like text baseline (HTML to text conversion), and SOM JSON. All evaluations read from the cached files to avoid content drift across formats.

The three representations are intended to capture common real-world agent input pipelines:

- **HTML:** full page response body.
- **Markdown baseline:** a compact text representation suitable for quick extraction tasks.
- **SOM:** a compact structured representation suitable for UI-aware tasks.

4.3 Task design principles

Tasks are written to cover both content-oriented extraction (for example, "What is the unemployment rate?") and structure-oriented understanding (for example, "What are the main navigation sections?"). Adversarial tasks emphasize noisy pages with heavy chrome or mixed content.

5 Experimental setup

5.1 Models and decoding

We evaluate two models: GPT-4o and Claude Sonnet 4. For both we use temperature 0.0 and max output tokens 2048.

5.2 Prompting

All conditions share the same system prompt, differing only in the page representation inserted into the user message. The prompt instructs the model to answer using only provided content and to return `"Not found on this page"` when the answer is absent.

5.3 Execution and missing data

Each task-format pair is executed three times. We log per-call input tokens, output tokens, and wall-clock latency.

Both model runs completed 900 calls (100 tasks x 3 formats x 3 runs). Due to missing cached corpus snapshots for a subset of URL IDs, 156 calls were skipped per model, leaving 744 successful calls per model. The distribution of successful calls is: HTML 267, markdown 267, SOM 210.

5.4 Metrics reported in this revision

- **Input tokens:** estimated for the prompt plus page content.

- **Output tokens:** estimated for the model response.
- **Latency:** wall-clock time from request to full response.

We also report bootstrap confidence intervals (CIs) for means.

6 Rubric scoring framework (accuracy and hallucination)

WebTaskBench is designed to support rubric-scored task accuracy and hallucination analysis. We define the scoring contract for each answer type.

Exact. Normalize whitespace and punctuation. Score 1 if the response matches an expected value under a task-specific tolerance (for example, numeric tolerance, unit normalization, or fuzzy match for headlines).

List and ordered list. Extract items and score using precision, recall, and F1 against a gold list. For a predicted set P and gold set G, precision $= |P \cap G|/|P|$, recall $= |P \cap G|/|G|$, and $F1 = 2PR/(P + R)$.

Table. Parse rows and columns into key-value pairs and compute micro-averaged F1 over expected cells.

Hierarchy. Treat the output as a tree and score depth and completeness. In practice we compute edge-F1 on (parent, child) pairs.

Summaries. Use human evaluation on a 1 to 5 scale for factual accuracy, completeness, and absence of unsupported claims.

Hallucination rate. Annotate each factual claim in a response as supported, unsupported, contradicted, or fabricated relative to the source representation. Hallucination rate is the fraction of claims that are not supported.

This revision reports the framework but does not yet include gold labels and human scoring results. We include these details to make the benchmark reproducible and to specify exactly how accuracy and hallucination metrics will be computed.

7 Results

7.1 Input token consumption

Table 2 reports average input tokens by format with 95% bootstrap CIs. SOM reduces average input tokens by 4.0x relative to raw HTML.

Format	Mean input tokens	95% CI	HTML ratio
HTML	33,181	[31,165, 35,201]	1.0x
Markdown	4,542	[3,972, 5,127]	7.3x fewer
SOM	8,302	[7,516, 9,058]	4.0x fewer

Table 2: Average input tokens by format (successful calls).

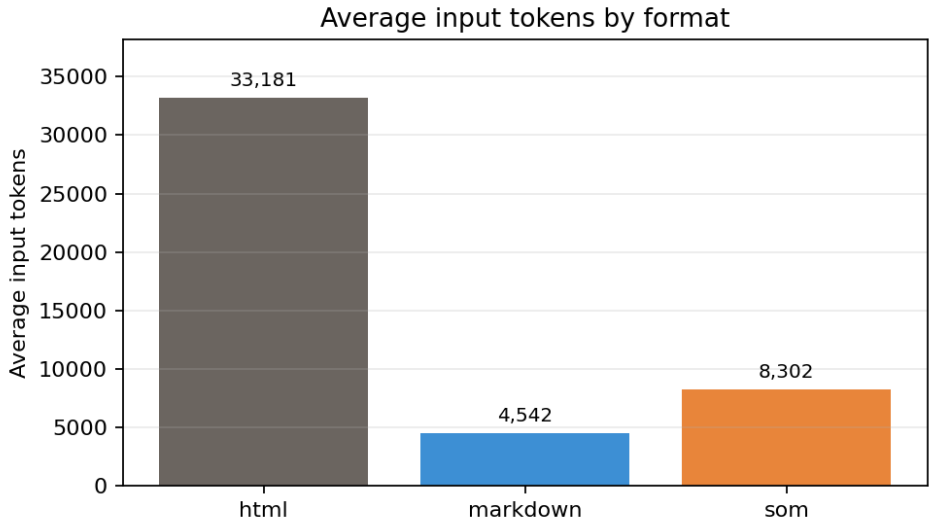

Figure 1: Average input tokens by format.

7.2 Category effects for token compression

Compression ratios vary by task category (Table 3). Navigation and adversarial tasks show the largest HTML-to-SOM ratios (5.4x and 6.0x).

Category	HTML	Markdown	SOM	HTML/SOM
Extraction	25,686	5,180	11,430	2.2x
Comparison	33,809	3,917	8,593	3.9x
Navigation	37,135	3,336	6,826	5.4x
Summarization	33,973	4,601	8,608	3.9x
Adversarial	34,893	5,749	5,846	6.0x

Table 3: Average input tokens by category and format (shared across models because prompt content is identical).

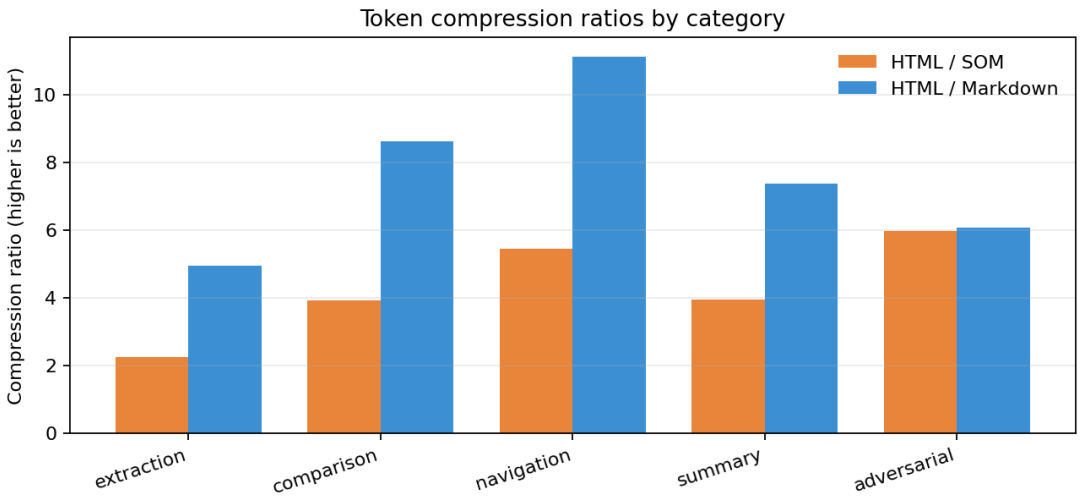

Figure 2: Token compression ratios by category.

7.3 Latency

Table 4 reports latency means with 95% bootstrap CIs. On both models, SOM is the fastest representation. On Claude Sonnet 4, SOM is substantially faster than markdown despite using more input tokens.

Model	Format	Mean latency (s)	95% CI
GPT-4o	HTML	2.74	[2.45, 3.06]
GPT-4o	Markdown	1.91	[1.65, 2.25]
GPT-4o	SOM	1.44	[1.30, 1.61]
Claude Sonnet 4	HTML	16.24	[13.99, 18.66]
Claude Sonnet 4	Markdown	25.17	[22.11, 28.35]
Claude Sonnet 4	SOM	8.51	[7.14, 9.98]

Table 4: Average latency by model and format (seconds).

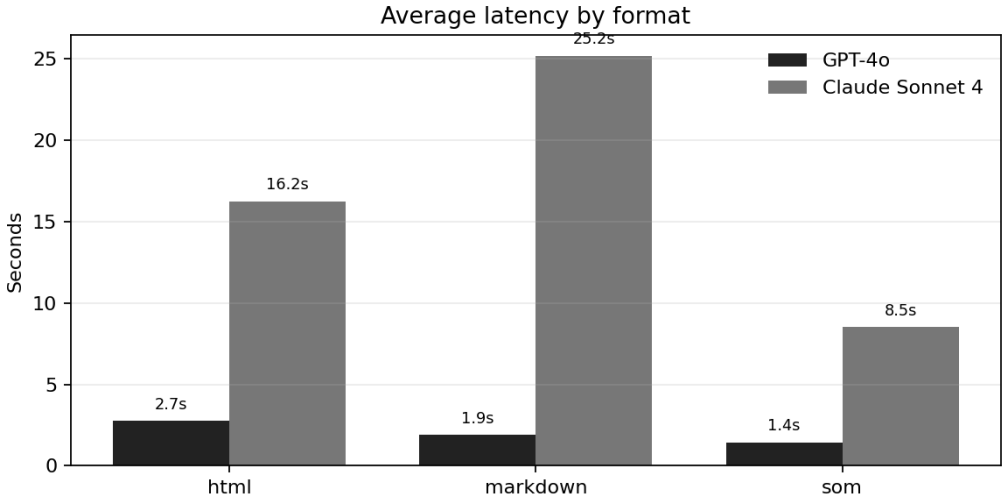

Figure 3: Average latency by format and model.

7.4 Latency by category (Claude)

Table 5 shows that the SOM latency advantage on Claude is consistent across categories.

Category	HTML	Markdown	SOM
Extraction	22.4	24.6	10.5
Comparison	20.3	33.6	10.5
Navigation	14.9	27.8	7.1
Summarization	13.1	19.6	7.5
Adversarial	10.5	19.4	7.0

Table 5: Average latency by category and format for Claude Sonnet 4 (seconds).

7.5 Output tokens and verbosity

Output tokens vary by model and representation. In our runs, Claude produced longer outputs than GPT-4o. SOM also tended to produce shorter outputs, likely because structured input

reduces the need for verbose re-statement of context.

8 Discussion

The results support three conclusions.

Format choice is not only about compression. SOM reduces input tokens relative to raw HTML, but the more surprising effect is latency. On Claude, SOM is far faster than markdown even though markdown is smaller.

Structure appears to reduce model work. A plausible mechanism is that SOM makes page regions and element roles explicit. This reduces the need for the model to reconstruct structure implicitly from text.

The largest gains appear in the hardest real-world cases. Navigation and adversarial pages are where agent systems typically struggle. The largest token compression ratios appear in these categories.

9 Threats to validity and limitations

Markdown baseline strength. The markdown representation used here is a simple HTML-to-text conversion baseline. Stronger baselines (for example, full readability extraction) should be included.

Missing data. Some URLs were missing complete cached snapshots across representations, leading to skipped calls. We treat these as missing data rather than model failures.

Scope of reported metrics. This revision reports cost and latency. Accuracy and hallucination outcomes require gold labels and human evaluation.

10 Future work

Future revisions will add rubric-scored accuracy for extraction, comparison, and navigation tasks; claim-level hallucination analysis; and human evaluation for summarization and adversarial tasks with inter-rater reliability. We also plan to expand coverage to more JS-heavy and login-gated pages and to add additional models, including open-weight systems.

11 Conclusion

WebTaskBench shows that format matters for agent browsing workloads. SOM offers a practical tradeoff: large token savings versus raw HTML while preserving structured meaning, and lower latency on both GPT-4o and Claude Sonnet 4.

References

[1] Zhou et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. 2023.

[2] Deng et al. Mind2Web: Towards a Generalist Agent for the Web. 2023.

[3] Duan et al. BrowserGym: A Benchmark for Web Agents. 2024.

[4] Nakano et al. WebGPT: Browser-assisted question-answering with human feedback. 2021.

[5] Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. 2022.

A URL list

URL ID	Domain
news-01	www.nytimes.com
news-02	www.bbc.com
news-03	www.reuters.com
news-04	arstechnica.com
news-05	www.theverge.com
news-06	apnews.com
news-07	www.washingtonpost.com
news-08	www.theguardian.com
news-09	www.cnn.com
news-10	techcrunch.com
ecom-01	www.amazon.com
ecom-02	www.target.com
ecom-03	www.ebay.com
ecom-04	www.etsy.com
ecom-05	www.bestbuy.com
ecom-06	www.zappos.com
ecom-07	www.homedepot.com
ecom-08	www.walmart.com
ecom-09	www.wayfair.com
ecom-10	www.costco.com
ref-01	en.wikipedia.org
ref-02	developer.mozilla.org
ref-03	github.com
ref-04	stackoverflow.com
ref-05	docs.python.org
ref-06	stripe.com
ref-07	docs.github.com
ref-08	kubernetes.io
ref-09	react.dev
ref-10	www.imdb.com
gov-01	www.sec.gov
gov-02	www.bls.gov
gov-03	www.gov.uk
gov-04	www.weather.gov
gov-05	www.usa.gov
social-01	news.ycombinator.com
social-02	www.reddit.com
social-03	lobste.rs
social-04	dev.to
social-05	discourse.org
saas-01	stripe.com
saas-02	linear.app
saas-03	vercel.com
saas-04	www.figma.com
saas-05	notion.so
adv-01	www.cnn.com
adv-02	www.forbes.com
adv-03	www.dailymail.co.uk
adv-04	www.allrecipes.com
adv-05	www.webmd.com

Table 6: WebTaskBench URL IDs and domains.

B Prompt template

```
System: You are a web research assistant. You will be given the content of a web
    page in [FORMAT] format. Answer the user's question based solely on the provided
     content. If the answer is not present in the content, say "Not found on this
    page."

User: [PAGE CONTENT]

Question: [TASK QUESTION]
```