

Is Firecrawl Worth It?

252 pages, 84 sites, 12 fetch pipelines and 20 Claude Code tasks: what Firecrawl buys in tokens, recall and dollars against the free tools you already have.

Drake Griffith

Co-founder and CTO, Actual Intelligence Labs

actualintelligencelabs.ai/research/is-firecrawl-worth-it

Abstract. We ran 252 URLs on 84 sites through 12 fetch pipelines, scored each output for tokens and content probes, and gave Claude Code 20 page-fact tasks in three setups. Firecrawl's markdown used 0.060x the tokens of raw HTML, a 94% cut, and caught 90% of content probes, the top recall in the test. It used 2.91x the tokens of trafilatura and 1.42x those of BeautifulSoup plain text, so against free extractors the credits buy recall and no token saving. On the Hobby plan a page cost \$0.0047, and against raw HTML Firecrawl pays for itself on any model that charges more than \$0.054 per million input tokens. Inside Claude Code, the Firecrawl MCP answered 19 of 20 tasks against 12 for the built-in WebFetch, at 4.06x the tokens and 3.13x the Claude dollars per task. Firecrawl's lead comes from ecommerce and JS-rendered pages; on docs, news, PDFs and small-business pages, BeautifulSoup's free plain text came within 6 points of its useful rate.

1. Short answers

Does Firecrawl cut tokens? Against raw HTML, yes. Its markdown used 0.060x the tokens (95% CI 0.040 to 0.090), a 94% cut. Against free extractors, no. It used 2.91x the tokens of trafilatura (CI 2.35 to 3.65), 1.42x the tokens of BeautifulSoup plain text (1.20 to 1.71), and about the same as Jina Reader (0.96x, 0.75 to 1.24).

Are the tokens content or metadata? About two thirds content. Firecrawl caught 90% of content probes, against 83% for raw HTML and Jina and 66% for trafilatura. Links and markdown wrapper took 37% of its tokens on the average page, and the full JSON that an API or MCP user receives raises that to 59%. If you strip links on your side, Firecrawl's tokens fall 43% and its useful-page rate drops 1.6 points.

Is it worth the cost? A page cost \$0.0047 on the Hobby plan (\$19 for 5,000 credits, all used, at 1.225 credits a page). Against raw HTML it saved about 87,000 tokens a page, so it pays for itself on any model that charges more than \$0.054 per million input tokens. Against trafilatura it saves no tokens. There you pay for recall: useful pages rise from 50% to 85%.

Does the MCP pay off inside Claude Code? The Firecrawl MCP answered 19 of 20 page-fact tasks; the built-in WebFetch answered 12. The MCP used 4.06x the tokens per task and 3.13x the Claude dollars, plus about 5 Firecrawl credits a task. Counting Sonnet's session input alone, the MCP used 7.46x as much, because WebFetch hands the page reading to Haiku. Its 28 tool definitions add about 25,000 tokens to each model call. Half of WebFetch's eight misses came from rate limits and a timeout that a retry might fix.

Verdict for a general user. Firecrawl earns its price in two cases: you send raw HTML or a browser dump to a model today, or your pages are ecommerce or JS-rendered and you run no browser of your own. On docs, news, PDFs and small-business pages, BeautifulSoup's free plain text came within 6 points of Firecrawl's useful rate on 74% to 101% of its tokens. In Claude Code, the MCP buys 7 more right answers in 20 for 4x the tokens. That trade pays if WebFetch fails on your sites. A cheaper setup, which we did not test, keeps WebFetch and calls Firecrawl when WebFetch fails.

2. Setup

URLs. 7 strata (docs, news and blogs, ecommerce, JS-rendered, bot-protected, PDF, small business) × 12 sites × 3 pages gave 252 URLs on 84 sites. We froze the list (urls.csv, SHA-256 32d6249b...3307) before any pipeline ran and swapped nothing after. The `urls` sheet of the page-study workbook holds the same rows.

Pipelines.

Code	What it is	Cost
B0	Plain GET, raw HTML	free
B0b	BeautifulSoup <code>get_text</code> on B0	free
B1off / B1on	trafilatura markdown on B0, links off / on	free
B2	Headless Chromium, then trafilatura	free
B3	Jina Reader, keyless	free
B4	Claude Code WebFetch (Haiku 5.5 digest of the page)	Claude tokens
T1	TinyFish Fetch, markdown	free
F1	Firecrawl /v2/ <code>scrape</code> , markdown, defaults	credits
F1json	The full F1 JSON response, as an API or MCP user receives it	same as F1
F1strip	F1 with link targets removed on the client	same as F1
F2	Firecrawl with <code>onlyMainContent</code> false (72 URLs)	credits

Measures. Tokens use OpenAI's o200k tokenizer. A calibration on 10 pages put Claude's count at 1.48x to 1.68x the o200k count, depending on format, and the ratios between pipelines moved under 10% in Claude tokens (F1/B1off 2.91 became 2.76; F1/B0 0.060 became 0.065).

A fixed, seeded rule drew up to 10 content probes per page (median 10): six-word windows of visible text outside header, nav, footer and aside, absent from the site's other two pages. It drew up to 5 boilerplate probes (median 5): text inside header, nav or footer, or text on all 3 of the site's pages. Both kinds come from a Playwright render, with B0's HTML as the fallback. A page counts as useful if the fetch succeeded and the output holds at least 80% of its content probes.

Tests treat the site as the unit, with equal weight per site. CIs come from a site bootstrap, p-values from sign-flip tests with Holm correction.

3. Tokens and signal, page by page

Pipeline	Geo-mean tokens	Useful rate	Content recall	Boilerplate leak	Link and wrap- per share
B0 raw HTML	26,798	.770	.83	.78	.82
F1json	2,559	.849	.90	.35	.59
B3 Jina	1,949	.766	.83	.51	.40
F1 Firecrawl	1,624	.849	.90	.35	.37
F1strip	931	.833	.90	.35	.01
B0b <code>get_text</code>	686	.762	.83	.77	.01
T1 TinyFish	582	.631	.73	.21	.01
B2 browser + trafilatura	540	.552	.67	.21	.03
B1off trafilatura	487	.504	.66	.19	.01
B4 WebFetch	181	.075	.08	.06	.05

Recall and leak count probes found, so a lower leak means less site chrome. B4's low scores reflect paraphrase: WebFetch hands Claude a small model's rewrite, and literal probes miss rewritten text (see Caveats).

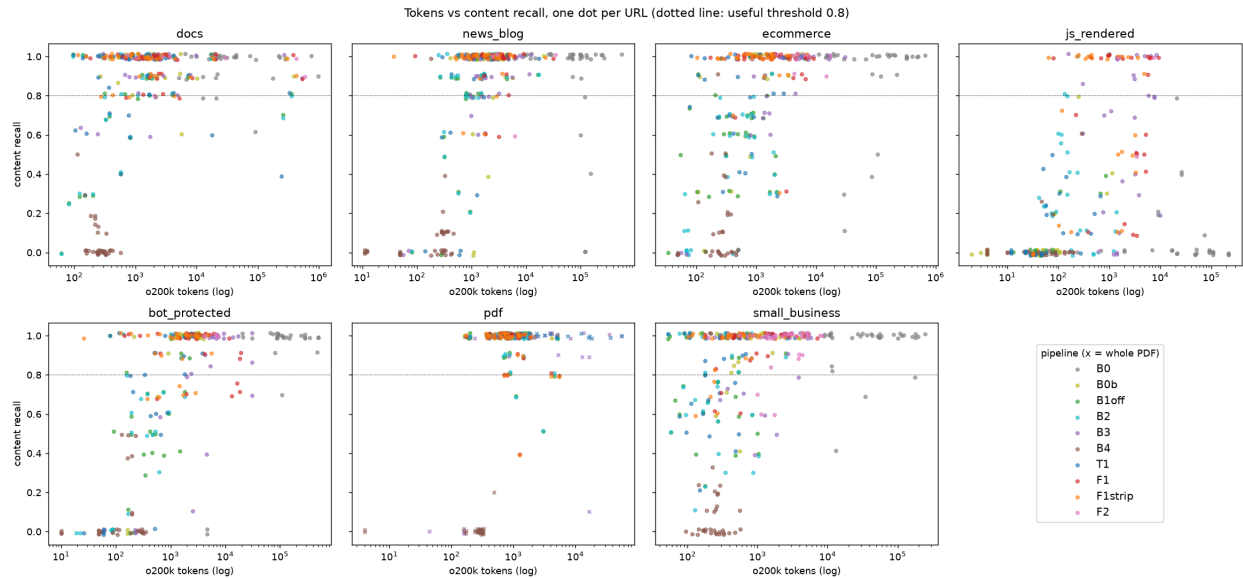


Figure 1. Tokens against content recall, by pipeline. One dot per URL; the dotted line marks the 0.8 useful threshold.

Useful rate by stratum.

Stratum	B0	B0b	B1off	B2	B3	Jina	F1	T1
docs	.97	.97	.72	.72		.92	1.00	.78
news_blog	.89	.89	.81	.86		.86	.94	.83
ecommerce	.81	.78	.17	.22		.69	.94	.61
js_rendered	.14	.03	.08	.28		.61	.50	.08
bot_protected	.72	.81	.33	.33		.58	.75	.56
pdf	.94	.94	.89	.89		.86	.89	.86
small_business	.92	.92	.53	.56		.83	.92	.69

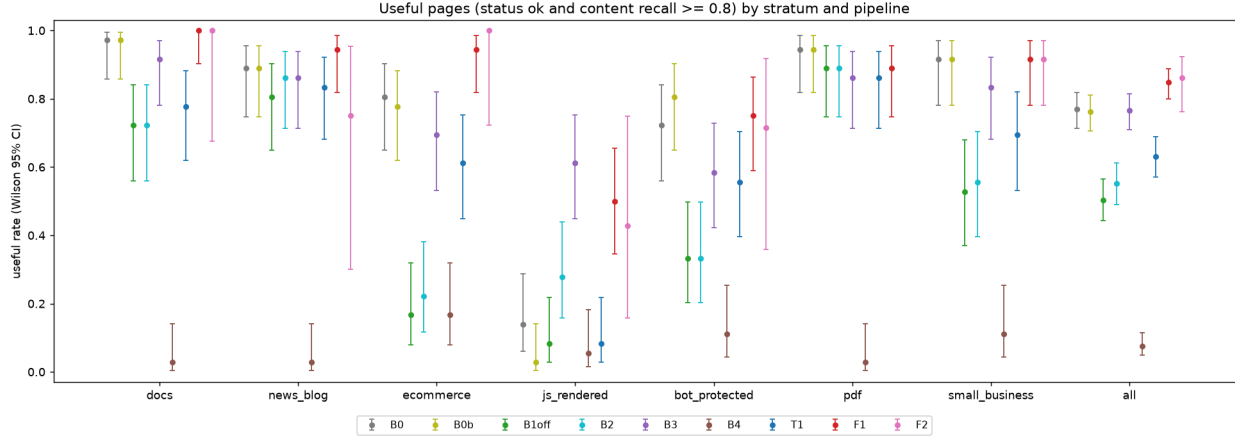


Figure 2. Useful rate by stratum and pipeline.

Firecrawl's lead over free routes comes from ecommerce and JS-rendered pages. Jina beat it on JS pages (.61 against .50). On bot-protected sites, plain `get_text` edged it (.81 against .75). Across the study Firecrawl refused 16 requests with HTTP 403 and charged nothing for them. On the 72 URLs we fetched both ways, the default `onlyMainContent` setting cut boilerplate leak from .92 (F2) to .39 (F1).

Overall, F1 beat B0b by 8.7 points of useful rate (CI 2.0 to 15.9). On the 231 URLs that had content probes, the gap shrinks to 5.7 points (CI about -1 to 12.9) and loses significance. F1 beat Jina by 8.3 points (CI 0.8 to 16.3; p 0.051 before Holm, 0.61 after).

4. Cost

Firecrawl charged 1 credit per HTML page and 1 per parsed PDF page (up to 3), 1.225 credits per successful page overall. The account's usage endpoint fell from 892 to 530 credits, which matches the 362 credits Firecrawl reported.

Plan use	\$ per page
Hobby, 5,000 of 5,000 credits used	\$0.0047
2,500 used	\$0.0093
1,000 used	\$0.0233

Against raw HTML, Firecrawl saved about 87,000 o200k tokens a page (CI 64,000 to 111,000), and half the pages saved under 40,000. Break-even sits at \$0.054 per million o200k tokens, about \$0.037 per million Claude tokens. Against trafilatura, Firecrawl added about 3,200 tokens a page, so the credits buy recall and no token saving.

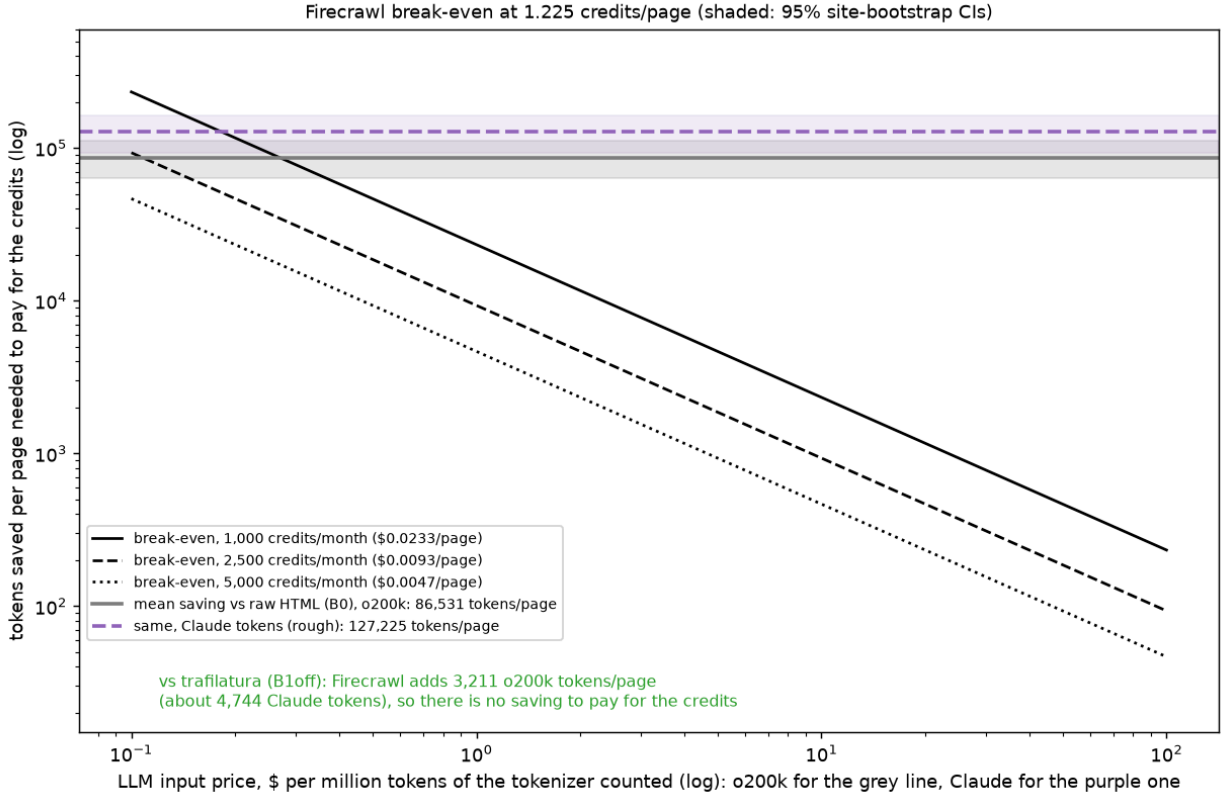


Figure 3. Break-even input price against raw HTML.

5. Inside a Claude Code session

20 page-fact tasks ran on Sonnet 5.5 in three setups. "Tokens" counts Claude's session input plus the tokens WebFetch's helper model spends.

Setup	Correct	Median tokens per task	Median Claude \$ per task	Firecrawl credits, 20 tasks
No web tools	0 / 20	4,410	\$0.0058	0
WebFetch	12 / 20	16,728	\$0.0096	0
Firecrawl MCP	19 / 20	62,081	\$0.0274	104 measured, 99 reported

- The MCP won 7 tasks that WebFetch lost and lost none (exact McNemar p 0.0156).
- Per task, the MCP used 4.06x WebFetch's tokens (CI 3.11 to 4.48) and 3.13x its Claude dollars (CI 2.41 to 4.75).
- Counting Sonnet's session input alone, the MCP used 7.46x the WebFetch setup's tokens (CI 5.62 to 7.82). WebFetch moves the page reading to Haiku, which pulls the combined ratio down to 4.06x.
- The account's usage endpoint showed 104 credits spent on the MCP tasks; Firecrawl's responses reported 99.

- The MCP's first model call carried 28,885 tokens against WebFetch's 3,894. The difference, about 25,000 tokens, is the MCP's 28 definitions, and it repeats on each call.
- The page study's 94% cut does not show up here. WebFetch already hands Claude a short digest (median 93 o200k tokens per tool result), while the MCP hands over the page (median 1,518).
- WebFetch's 8 misses: 4 transient (three HTTP 429s from Allbirds, Brooklinen and Tentree, one NPR timeout), 3 lasting (WebFetch blocks the Guardian, and two small-business pages returned a title or the word "Store" and nothing else), and 1 that both tools missed. On the lasting causes alone, the MCP leads 3 to 0, p 0.25.
- The miss both tools shared asked for a Cloudflare post's author: the MCP named Chetan Agrawal; the gold byline reads Matthew Gumport.

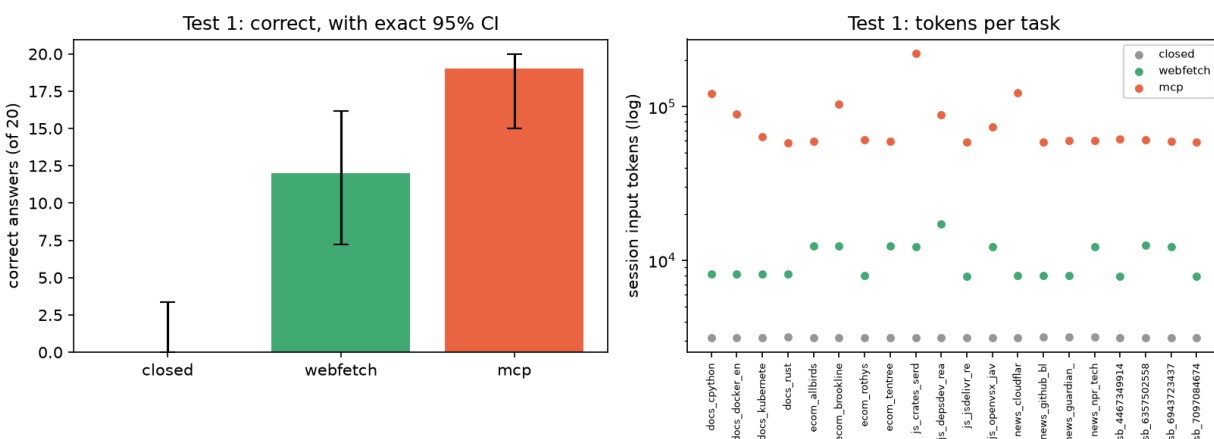


Figure 4. Claude Code test (labelled Test 1 in the chart): correct answers with exact 95% CIs, and session input tokens per task.

6. Caveats

- **Literal probes understate paraphrase.** A probe counts as found if all six words appear verbatim. Tools that rewrite text (WebFetch, any summarizer) score low even when the facts survive.
- **The pilot's JS probes were weak.** The pilot's JS page drew probes from one-word quote tags, which can match text elsewhere on a page. The pilot had 4 pages.
- **Claude wrote the rule and judged the spot-check.** In a sample of 605 probes, the spot-check judged 41 wrong (6.8%, Wilson CI 5.0% to 9.1%): 33 of 421 content probes and 8 of 184 boilerplate probes. Counting firm errors alone gives 29 (4.8%). 21 URLs had no content probes; for them, useful means the fetch succeeded.
- **The probe render shares Chromium with B2.** B2 reads the same rendered page the probes came from, which favours B2's recall.

- **Boilerplate probes rest on tags.** The rule finds chrome through header, nav and footer tags and through text repeated across a site's pages. Chrome built from plain divs escapes it, so leak figures may run low.
- **"Useful" ignores leak and structure.** Raw HTML scores well on useful rate while 82% of its tokens sit in tags and links.
- **Claude token counts are rough.** No API key was available, so we measured on the subscription: input tokens with a page minus input tokens without it, on Haiku 5.5, over 10 pages.
- **The MCP's win in Claude Code depends on transient failures.** A retry policy for WebFetch's 429s could close half the gap.
- **The stratum mix sets the averages.** Each stratum carries equal weight. A user who reads docs and news will see a smaller Firecrawl advantage than the overall numbers show; a user who scrapes shops will see a larger one.
- **Credits.** Firecrawl went from 996 to 530 credits (104 in the Claude Code test, 362 in the page study).

7. Data

Two Excel workbooks hold the rows behind each table and chart in this report. Each opens on a README sheet that describes its sheets.

- **Page study** ([is-firecrawl-worth-it-page-study.xlsx](#)): the frozen URL list, per-URL scores for each pipeline, the stratum table, the confirmatory and exploratory tests, token ratios by stratum, partially pooled useful rates, savings concentration, cost break-even, the Claude token calibration, the content and boilerplate probes, the probe spot-check and the fetch log.
- **Claude Code test** ([is-firecrawl-worth-it-claude-code-test.xlsx](#)): the 60 runs (20 tasks in 3 setups) with gold answers and grading, totals per setup and the paired tests.

Email addresses in both workbooks keep their domain only, and phone numbers keep their area code only.