ChatGPT

Crawling Benchmark Analysis and Score Breakdown

Score Calculation Methodology

The **quality score** in the benchmark is a composite metric (scaled 0 to 1) that was calculated as a weighted sum of several performance factors. Each factor corresponds to specific result columns in the data. The breakdown is as follows:

- **Fetch Success (30% weight):** Indicates if the page was fetched successfully (`fetch_success_1_0` = 1 or 0). A successful HTTP fetch contributes 0.30 to the score, while a failure contributes 0 for this portion.
- **Content Sufficiency (25% weight):** Checks if the extracted main content meets or exceeds the expected minimum length (`main_content_chars` vs. `expected_min_chars`). If yes, `content_ok_1_0` = 1, contributing 0.25 to the score. If the content is too short or missing, this contributes 0.
- **Metadata Presence (15% weight):** Measures whether key metadata (especially the publication date, and also title/author if applicable) were extracted. This is reflected by a meta "hit rate" (`meta_hit_rate`) or binary flags (`title_present_1_0`, `date_present_1_0`, etc.). If the important metadata (notably the date, since all test pages were expected to have one) was captured, full 0.15 is earned. Partial metadata capture may give partial credit (e.g. some tools might extract the title but not the date).
- **Content Cleanliness (15% weight):** Assesses the proportion of noise (boilerplate, nav elements, ads, etc.) in the content. This uses `noise_estimate` (the fraction of content characters deemed noise) to compute a cleanliness score = 1 – noise_estimate. If a page's content was very clean (little noise), it earns up to 0.15. Heavily noisy content reduces this portion of the score.
- **PDF Parsing (10% weight):** For PDF documents, this checks if the PDF was parsed adequately. If the URL was a PDF, getting at least a minimum number of pages parsed (parameter `pdf_min_pages` = 2) yields `pdf_ok_1_0` = 1. If the PDF couldn't be parsed or had too few pages, this portion is 0. (For non-PDF URLs, this factor defaults to 1, granting the full 0.10 since PDF parsing isn't applicable.)
- **JavaScript Rendering (5% weight):** For pages that required JS rendering (marked by `expected_js_required = Y`), this checks if the tool successfully rendered the page/content with JS. If yes, `js_ok_1_0` = 1, contributing 0.05. If the tool failed to render needed JS content, this portion is 0. (For pages not requiring JS, this factor is automatically satisfied, giving the full weight by default.)

All these weighted components sum up to the total **quality score** for each run. For example, if a tool successfully fetched a page, met content length expectations, captured the date/title, produced clean output, and the page didn't require JS or was a non-PDF, it would score a perfect 1.0 (100%). In contrast, missing any of these aspects subtracts the corresponding weight. *(Note: The benchmark also tracked an efficiency score separately, factoring in latency and cost, but the primary "score" discussed here refers to the quality score unless otherwise noted.)*

Performance Comparison Across Tools

Three different crawling solutions (entities) were evaluated in the dataset: **Tavily**, **Serper**, and **Parallel**. Each tool was run on a common set of benchmark URLs (18–19 URLs per tool) and their performance was measured by the quality score and related metrics. Table 1 below summarizes key performance metrics for each tool:

Table 1. Overall Performance by Tool

Tool	Runs	Success Rate	Avg. Quality Score	Avg. Efficiency Score	Content_OK Rate	Meta Hit Rate	Fail Rate
Parallel	18	100%	**0.760**	0.502	88.9%	62.96%	0%
Tavily	19	100%	0.653	0.320	63.2%	36.84%	0%
Serper	18	83.3%	0.572	0.357	38.9%	46.30%	16.7%

Cited from the benchmark Dashboard data. "Content_OK Rate" is the proportion of runs where extracted content met the expected minimum length; "Meta Hit Rate" is the fraction of runs capturing key metadata (title/date/author).

From the above results, **Parallel emerges as the top performer** in terms of quality. It achieved the highest average quality score (~0.76 out of 1.0), indicating that it consistently fulfilled most of the content extraction criteria. Parallel also had a perfect 100% success rate (no fetch failures) and the highest content sufficiency rate at ~88.9% (meaning in nearly all its runs it captured at least the expected amount of content). In metadata extraction it led as well, with ~63% meta hit rate, suggesting it often pulled key info like dates successfully. Notably, Parallel accomplished this while maintaining a reasonably good efficiency (average efficiency score ~0.502, corresponding to moderate speed and low resource usage).

Tavily showed **moderate performance** in comparison. Its average quality score was ~0.653, second-best among the tools. Tavily also had a 100% success rate (no complete failures), which is a positive. However, it met the content-length threshold in about 63% of cases, lagging behind Parallel on content extraction success. Tavily's metadata capture rate (~37%) was the lowest of the three, indicating it frequently missed some meta information (particularly the date) in many pages. On the positive side, Tavily was the only tool that managed any JavaScript-required pages successfully – it has a JS success rate of ~66.7% for the JS-heavy pages, whereas the others captured none of those (Parallel and Serper have 0% `js_ok_rate`). This suggests Tavily attempted to render JavaScript content on some pages, giving it an edge on those specific cases. In terms of efficiency, Tavily was the slowest/most resource-intensive (avg latency ~4.45 s and higher credit usage), reflected in the lowest efficiency score (~0.320). This implies Tavily's approach, while achieving decent quality, might be heavier or slower, which could be a trade-off for its ability to handle JS.

Serper had the **lowest overall quality performance** in this benchmark. Its average quality score was ~0.572, significantly lower than Parallel's, indicating more frequent issues. Serper suffered from some outright failures – about 16.7% of its runs failed to fetch or extract content at all (reflected by its 83.3% success rate and corresponding 16.7% fail rate). These failures dragged down its average score. Even when

it succeeded in fetching, Serper often extracted insufficient content (content OK in only ~38.9% of cases) – the lowest content success rate among the tools. This suggests that on many pages, Serper's output was missing large portions of the main content, which severely impacted its quality score. On a positive note, Serper's metadata hit rate (~46%) was better than Tavily's, indicating it sometimes captured dates/titles when it did get content, but this did not compensate for missing core content. Unlike Tavily, Serper did **not** handle any of the JS-required pages successfully (`js_ok_rate` 0%), which contributed to its failures (e.g. it completely failed on the heavy JS page, yielding a very low score on that run). One advantage of Serper was efficiency: it was the fastest and most lightweight tool (average latency ~1.5 s, lowest credits used), giving it the highest efficiency score (~0.357 among these) after Parallel. In summary, Serper traded off quality for speed – it completed crawls quickly, but often with incomplete results or outright misses.

In terms of **best and worst performers**, **Parallel performed the best** overall according to the quality score, excelling in reliability and content extraction. **Serper was the weakest** on quality due to its inconsistency and failures, despite its speed. Tavily was in the middle, leaning closer to Parallel in quality success but with some shortcomings (especially slower performance and lower metadata capture).

Score Distribution and Threshold Analysis

To better understand the range and groupings of scores, we can analyze how each tool's individual run scores are distributed. A **threshold-based analysis** reveals natural performance groupings into high, medium, and low quality outcomes:

- **High-Performing Runs (Score > 0.8):** These are cases where the crawler nearly met all criteria (successful fetch, sufficient content, metadata, etc.). Across all tools, a score above 0.8 indicates an excellent extraction result. In the benchmark, Parallel had the most high-performing runs – about one-third of its pages scored above 0.8. Tavily achieved a high score >0.8 on only a couple of pages (~11% of its runs), and Serper on a few (~17%). This shows that **Parallel consistently hit the high-quality mark** far more often than the others.
- **Medium Performance (Score ~0.5 to 0.8):** Scores in this range indicate partial success – perhaps the content was fetched and mostly okay but some elements were missing (e.g., maybe the date wasn't captured or content was slightly under the threshold). The majority of runs for Tavily fell into this medium band (around 53% of Tavily's scores were 0.5–0.8) and about 39% of Serper's runs were medium. Parallel had the bulk of its runs here (about two-thirds between 0.55 and 0.8), with many clustering in the 0.7–0. Eight range. The medium segment is where **Tavily and Parallel mostly operated** (aside from Parallel's many highs), indicating generally solid but not perfect outcomes.
- **Low-Performing Runs (Score < 0.5):** Scores below 0.5 signify poor outcomes – significant content was missing or the crawl failed in some way. **Serper had a large portion of low scores** (roughly 44% of its runs scored under 0.5), aligning with the earlier observation of its frequent failures and incomplete extractions. Tavily also had a non-trivial number of low-performing cases (~37% of Tavily's runs under 0.5), likely pages where it may have missed the content length target or other key info. Notably, **Parallel had zero runs in this low category** – it never dropped below a ~0.55 score in any test. This absence of low scores for Parallel underscores its consistency: it **never had a truly bad extraction** in the benchmark, whereas the other tools did on multiple occasions.

Looking at these groupings, there's a **natural cutoff around the 0.5 score mark** separating "acceptable" vs "unacceptable" outcomes. A score of 0.5 corresponds roughly to meeting about half the criteria (for instance, a page was fetched but either content or meta was missing). Both Tavily and Serper had several

runs below 0.5, which could be considered clear failures or poor results that likely need attention. On the upper end, scores near 0.8–1.0 denote very successful runs; Parallel frequently operated in this range, whereas it was relatively rare for Tavily and Serper. We can thus classify Parallel as a **consistently high-performing crawler**, Tavily as **moderate with occasional highs and some lows**, and Serper as **highly variable with many lows**. These thresholds (around 0.5 and 0.8) serve as natural breakpoints to group the tools' performance into low, medium, and high quality tiers.

Rankings of Tools by Score

Based on the average quality scores and the consistency of results, the tools can be **ranked** as follows:

1. **Parallel** – *Rank 1.* **Highest quality** performer. Parallel tops the ranking with an average quality score of ~0.76. It not only scored highest on average, but also had 100% success rate and no low-quality runs, indicating robust and reliable performance across all test pages.
2. **Tavily** – *Rank 2.* **Moderate quality** performance. Tavily's average quality score (~0.65) was solid but notably lower than Parallel's. It had no complete failures, which is good, but it struggled with content and metadata on a number of pages, keeping many of its scores in the mid-range. Tavily is the runner-up, performing reasonably well overall but not reaching Parallel's level of completeness.
3. **Serper** – *Rank 3.* **Lowest quality** performance. Serper ranked last with the lowest average score (~0.57). It was the only tool to suffer multiple outright failures, and it frequently delivered incomplete content. While it had a few successful runs, its inconsistency and the large fraction of poor outcomes put it firmly in third place in terms of quality.

It's worth noting that this ranking is strictly by the quality score. If we considered other factors like speed or efficiency, the order might differ (since Serper was fastest, and Tavily slowest). However, in terms of the benchmark's primary metric – extraction quality – **Parallel > Tavily > Serper** is the clear order from best to worst.

Key Findings and Insights for Decision-Making

- **Parallel is the Best All-Around Performer:** It achieved the highest quality scores with no failures. Parallel consistently extracted the required content and metadata from every page, making it a reliable choice when completeness is critical. Its runs never fell into the "low" performance category. This suggests that if **maximizing extraction quality and reliability** is the priority, Parallel is the top candidate.

- **Tavily Balances Quality with JS Capability (at a Cost):** Tavily delivered decent quality (second-place) and was notably capable of handling JavaScript-heavy pages that others could not, thanks to its partial JS rendering success. This makes Tavily valuable for content that requires client-side rendering. However, this came with trade-offs: it had the slowest performance and higher resource usage (lower efficiency score) and still missed metadata more often. For **use-cases involving dynamic pages**, Tavily might be a good option, but be mindful of its speed and ensure that missing metadata can be tolerated or mitigated.

- **Serper Prioritizes Speed Over Completeness:** Serper stood out for its fast crawl times and low resource usage, but this seems to have come at the expense of quality. Its lower scores and higher failure rate indicate that it often did not get the full content or any content at all in some cases. In

scenarios where **speed is essential and some content loss is acceptable**, Serper could be useful. However, if the task requires reliable data extraction from each page, Serper's performance is too erratic – many pages will be incomplete or require re-crawling. Any deployment of Serper would need a mitigation plan (like retries or supplementary methods for pages it fails).

- **Content Sufficiency is a Key Differentiator:** The metric for capturing enough content (`content_ok_rate`) exposed major differences between the tools. Parallel's ability to meet the content threshold ~89% of the time was significantly higher than Tavily's ~63% and Serper's ~39%. This indicates that **Parallel not only succeeded in fetching pages, but also extracted nearly all the main content**, whereas the others frequently left out large portions. For decision-makers, this is crucial – a crawler that returns incomplete data can undermine the usefulness of the result. Thus, content sufficiency (and by extension, the quality score) should be weighed heavily when choosing a solution.

- **No Single Tool Excels in Every Aspect:** Each tool had strengths and weaknesses. Parallel excelled in quality and consistency but did not attempt JS-heavy rendering (it scored 0 on JS-required pages). Tavily handled JS and had no outright failures, but was slower and not as thorough in content/meta extraction. Serper was very fast and lightweight but unreliable in data quality. This suggests an opportunity for **optimization or a hybrid approach**: for example, one might use Parallel as the default crawler for most pages, but switch to Tavily for pages known to require heavy JS, to combine the strengths of both. Alternatively, improving Tavily's content extraction and metadata parsing, or Parallel's JS rendering capability, would yield a more well-rounded tool. These insights can guide where to focus development: e.g., boosting Tavily's parsing accuracy, or adding JS rendering to Parallel's toolkit would address their current shortcomings.

In conclusion, the **quality score** provides a clear composite measure to evaluate crawler performance across multiple dimensions (success, content, meta, cleanliness, etc.). By analyzing this score, we identified Parallel as the high-quality leader, Tavily as a solid middle performer with a niche in JS content, and Serper as a fast but risky option. The thresholds in score distribution (high vs. low performers) further highlighted Parallel's reliability versus the inconsistency of the others. These findings can inform decisions on which crawling solution to deploy and where to target improvements – ensuring that the chosen tool meets the desired balance of completeness and efficiency for the task at hand.
