



WHITE PAPER

Why Not Just Use Claude?

Why Institutional Data Programs Require More Than an LLM for Reliable, Repeatable,
Verifiable Web-Scraping Operations at Scale

A Sequentum White Paper

For Investment Banks, Retailers, and Value-Added Data Resellers Evaluating AI-Assisted Web Data Extraction

September 2026

This document is provided for general informational and educational purposes only. It does not constitute legal, regulatory, compliance, or investment advice. Organizations should consult qualified counsel and compliance professionals regarding their specific obligations under securities, data protection, and other applicable laws before relying on any web-scraped or AI-assisted data process. Product and architecture descriptions reflect Sequentum's platform as of the date of publication; illustrative scenarios in this paper are hypothetical composites built from common patterns in each industry and do not describe any specific customer's actual engagement, data, or workflow.

Contents

Contents	3
Executive Summary	4
1. The New AI-Assisted Scraping Landscape	4
2. The Determinism Problem	5
2.1 What “deterministic” means, and why it is assumed by default	5
2.2 Same input, different output	5
2.3 What this means inside a scraping pipeline	6
2.4 Why agent harnesses and frameworks don't change this	7
3. The Access Problem: Anti-Bot Defenses	8
3.1 Getting blocked before you ever reach the data	8
3.2 Why a live LLM session gets flagged	8
3.3 A second, independent failure mode	9
4. Why This Matters Even More for Institutional Users	9
4.1 Investment banks and asset managers	9
4.2 Retailers and consumer brands	10
4.3 Value-added data resellers	10
5. The Ideal Architecture: AI-Assisted Build, Deterministic Run	11
5.1 Natural language in, working prototype out	11
5.2 Additional benefits: human refinement, full control	11
5.3 Deterministic execution at run time	11
5.4 Governance, monitoring, and change management	12
5.5 Operating reliably against anti-bot defenses	12
6. Illustrative Use Cases	13
6.1 Investment Bank: Alternative Data for Equity Research	13
6.2 Retailer: MAP and Competitive Price Monitoring at Scale	14
6.3 Value-Added Data Reseller: Contractual Data Feeds for Enterprise Customers	14
7. Risk and Compliance Considerations	15
7.1 Recordkeeping regimes	15
7.2 Material non-public information exposure	15
7.3 Data protection and personal information	15
7.4 Terms of service, copyright, and site-load diligence	15
8. Conclusion and Recommendations	16
Recommendations	16
References	17

Executive Summary

Generative AI has made it possible to describe a web-scraping task in plain English and have a working prototype minutes later. That capability is valuable in its own right. It collapses weeks of engineering effort into an afternoon. But an AI-assisted prototype and a production data pipeline are not the same thing, and the gap between them matters enormously to institutions that depend on scraped data to inform investment decisions, set pricing, and retain customers.

Large language models, including Claude and every other frontier model, are non-deterministic by construction. The same prompt sent to the same model can produce different output from one run to the next. Although that variability is a curiosity for a chatbot, it is disqualifying for a system that extracts prices, financial disclosures, or inventory levels that feed a trading model, a compliance filing, or a customer-facing data feed.

One way to resolve this tension is to use an LLM for the part of the job it's actually good at: interpreting a natural-language description and assembling a working scraping agent. From there, hand the result off to a deterministic execution engine for everything that follows. Once built, the agent can be versioned, tested, and audited. Because it isn't re-imagined by the LLM each time, it runs the same way, produces the same schema, and can be inspected, refined, and certified the way any other production system can. That gives you the speed of AI-assisted development without the reliability problems of using an LLM as the extraction engine itself.

This paper explains why that distinction is a structural requirement for investment banks, retailers, and value-added data resellers; walks through illustrative scenarios in each sector; and lays out the specific risks organizations take on when they run production data operations directly on top of a raw LLM rather than on a deterministic platform built around one.

1. The New AI-Assisted Scraping Landscape

Over the past two years, three overlapping approaches have emerged as teams try to put LLMs to work on web data extraction. The first is direct prompting, where a user — from an analyst, engineer, or researcher to a developer or founder testing a theory — pastes raw HTML or a screenshot into a chat interface and asks the model to pull out the fields they need. The second is the LLM-powered browsing agent: a model is given tools to click, scroll, and read a live page, and asked to complete an extraction task end to end, run after run. The third, and the one this paper is centrally concerned with, is the AI-assisted agent builder: a natural-language description is used to generate a defined, inspectable set of navigation and extraction rules which is then executed independently of the model that created it.

The appeal of the first two patterns is obvious. They require no code or site-specific engineering, and you get a working result on the first try more often than not. That appeal is exactly why so many teams have quietly started routing production data collection through a raw LLM call, without stopping to ask a harder question: what happens on run two, run two hundred, and run two thousand, when the page hasn't changed but the model's answer has?

This paper argues that using an LLM as the extraction runtime itself is acceptable for one-off research, exploration, and ad hoc analysis, but not for any process an institution intends to run repeatedly, rely on operationally, or stand behind contractually or to a regulator. On the other hand, the third pattern, where AI functions as the developer not the runtime engine, is the one that scales safely.

2. The Determinism Problem

2.1 What “deterministic” means, and why it is assumed by default

A deterministic process produces the same output every time it's given the same input. Traditional software, including a conventional web-scraping script, is deterministic almost by default: the same page structure, parsed by the same rule, yields the same extracted value today and a year from now, unless someone changes the rule or the page itself changes. That property is so basic to how software behaves that it's easy to take for granted, and easy to assume an AI-built system will inherit it. It doesn't.

2.2 Same input, different output

Ask a large language model the same question twice, using the exact same prompt, and there is no guarantee you will get the same answer back. This is true of Claude as well as every other frontier model. It holds regardless of how carefully the prompt is written, and it does not go away just because the desired output format was specified in advance.

Independent research on ChatGPT's code-generation behavior gives a sense of scale: sending the identical prompt five times to the same model produced completely different outcomes on anywhere from 18% to 76% of test problems, depending on the benchmark and settings used, including in the “low-randomness” settings most people assume will produce a consistent answer.¹ Follow-on research has since confirmed the same instability regardless of which model, vendor, or infrastructure is involved.²

What we saw when we tested this

We ran this test ourselves against a live page on Kendra Scott's website: the product listing for its Alice Color Beaded 14k Gold-Filled Chain Bracelet. We asked an LLM to extract a fixed schema, including metal and stone, three separate times, using the exact same page content each time. All three runs agreed on the metal: Gold-Filled. They didn't agree on the stone. The only clue on the page is a color option labeled

“Mixed Blue Enamel.” Two of the three runs correctly read that as not a gemstone at all and returned “None.” The third run called it “Enamel” and reported it as the stone. Same input, same schema, three runs, two different answers to whether this product even has a stone.

That inconsistency is the focus problem of this paper, and it is easy to reproduce yourself: run the same extraction request against the same page a few times and compare the answers. For a chatbot answering a one-off question, that variability barely registers, but for a system meant to produce the same structured record for the same product, every time, it is disqualifying. And no amount of prompt engineering makes it go away, because the inconsistency is in the model's answer, not in the prompt.

2.3 What this means inside a scraping pipeline

Translated into operational terms, using a raw LLM as the extraction engine, rather than as a one-time developer of a fixed extraction routine, exposes a production pipeline to problems that are difficult or impossible to fully engineer around:

- Silent field drift: the same unchanged page can yield a different set of extracted fields, a differently formatted number, or a subtly reworded text field from one run to the next, with no error thrown and no obvious signal that anything changed. That's exactly what happened with the stone field in the Kendra Scott example above.
- Fabrication under ambiguity: when a page is malformed, paginated unusually, or contains data the model wasn't confident about, an LLM will frequently produce a plausible-looking value rather than a clear failure. That's the well-documented “hallucination” behavior of generative models, and it's far more dangerous in a data pipeline than a hard error, because it doesn't trigger any exception-handling path.
- No stable baseline for change detection: one of the most operationally valuable properties of a deterministic scraper is that when the extracted output changes, that change means something, usually that the target site changed. When the extraction step itself is a source of randomness, teams lose the ability to distinguish “the website changed” from “the model answered differently this time.” That's exactly the signal a data operations team needs to trust and act on.
- Unpredictable cost and latency at scale: model API calls at production volume introduce per-record latency and token cost that compounds across large catalogs, and any mitigation strategy, such as running each extraction multiple times and reconciling the results, multiplies both.
- Version and model drift: providers periodically update or deprecate model versions. A pipeline built directly on a live model call is exposed to behavior changes on a timeline the pipeline owner does not control, requiring re-validation of production behavior with no code change of their own.

The core distinction

A well-engineered scraping program answers the question “what is on this page, according to this fixed rule?” every single time. A live LLM call answers the question “what does the model currently believe is on this page?” That second answer may change for reasons that have nothing to do with the page itself.

2.4 Why agent harnesses and frameworks don't change this

A natural objection at this point is that the extraction call itself doesn't have to be a bare, unmanaged prompt. A growing category of “agent harnesses” and orchestration frameworks (general-purpose runtimes that wrap an LLM with session management, tool access, memory, and retry logic) are increasingly used to give AI agents production-style structure. But it's worth being precise about what these tools actually promise, because it isn't what this objection assumes.

Frameworks built for adaptive web automation are explicitly designed around the opposite of reproducibility. OpenClaw's own browser-harness documentation, for example, describes an agent that uses “no framework, no pre-written recipes, no fixed selectors” and that “self-heals when a step fails.”³ In practice, that means it adapts its approach dynamically each time a page behaves unexpectedly. That's a sensible design goal for an interactive task on an unpredictable site, but it isn't a determinism guarantee, and the harness's own documentation doesn't claim to be one. Self-healing means the agent is deliberately free to take a different path when something looks off. That's another name for the run-to-run variability Section 2.3 describes.

A more serious version of the objection asks: what if a team configures the harness to invoke the LLM only once per source, capture whatever selector or extraction plan it produces, and replay that fixed artifact on every subsequent run without calling the model again? That configuration would, in fact, produce deterministic output. But at the moment a team does that, the harness is no longer doing the extraction. A pinned, cached, non-re-invoked artifact is exactly the pattern this paper describes as the alternative to raw LLM extraction: it's just been assembled ad hoc, source by source, rather than built for it. The question that matters at that point is no longer whether determinism is achievable; the team has just demonstrated that it is. The real question is who versions that artifact, tests it before it reaches production, gets notified when it silently stops matching the page, and can show a regulator or a client exactly what ran and when. That's the operational and governance layer a general-purpose harness isn't built to provide, and that a hand-assembled collection of pinned scripts has to reinvent, one source at a time.

Nor would a different harness, a newer framework, or a future version of any of these tools change the underlying picture. The disagreement in the Kendra Scott example wasn't caused by a badly written prompt or a limitation of one specific browsing tool. It's what happens whenever a model is asked to make the same judgment call fresh, on its own, every time it runs. A harness can add memory, tools, scheduling, and retries around that judgment call, but it doesn't change the judgment call itself. Until the model stops

being asked to decide the answer fresh on every run, the inconsistency remains, no matter how well engineered the layer built around it is.

3. The Access Problem: Anti-Bot Defenses

3.1 Getting blocked before you ever reach the data

The determinism problem assumes a model actually gets a clean look at the page. In practice, that assumption often fails before extraction even starts. Most commercial websites, and virtually every site with pricing, inventory, or financial data worth scraping, run some form of bot detection, from Cloudflare and DataDome to a growing list of specialized vendors. A raw LLM browsing session, or an agent harness built on top of one, is not exempt from these defenses. If anything, it's an easier target than a purpose-built scraper, because it tends to look and behave like automation in ways that are simple to detect.

3.2 Why a live LLM session gets flagged

Anti-bot systems don't need to know a request came from an LLM specifically. They only need to notice that it doesn't look like an ordinary browser session, and a live model-driven browsing session gives them several ways to notice:

- Network and IP reputation: LLM API calls and browser-automation sandboxes typically run from cloud data-center IP ranges with known, poor reputations, rather than the residential or mobile IP ranges a real visitor uses. Detection systems check this before they check anything else.
- TLS and connection fingerprinting: the underlying HTTP client or automated browser stack an agent runs on produces a distinctive, stable handshake signature, commonly called a JA3 or JA4 fingerprint, that differs from an ordinary consumer browser and is easy to flag at the network level, before a single page even loads.⁴
- Browser and device fingerprinting: JavaScript-accessible signals, such as user agent string, screen resolution, installed fonts, and how the browser reports its own automation flags, reliably distinguish an automated session from a human one. In independent research, this single layer alone identified automated browsing agents with over 93% accuracy.⁴
- Behavioral analysis: no mouse movement, no scrolling, uniform timing between actions, and requests that arrive faster or more regularly than a person would produce are all classic markers of an automated session.
- Challenge-response failures: CAPTCHAs, invisible proof-of-work checks, and interactive challenges are specifically designed to stop a script or a model in its tracks. Reliably solving them isn't something a browsing agent can promise to do on every run.

- Explicit, policy-level blocking: major providers now let a site owner block AI agents and crawlers outright, by user agent, by declared purpose, or by default. Cloudflare, for one, now offers site owners a single control to block known AI bots, scrapers, and crawlers.

Attempts to mask any of this, such as spoofing a user agent, faking headers, or running a stealth browser profile, don't reliably solve the problem either. Research on this specific question found that stealth and anti-detection tactics often make an automated session more identifiable, not less, because the inconsistencies they introduce become a signal of their own.⁴

3.3 A second, independent failure mode

This is a different problem from the one Section 2 describes, and it compounds it rather than replacing it. A determinism failure means the model read the page and gave two different answers. A blocking failure means the model has never read the real page at all. It got a CAPTCHA page, an error page, or a deliberately distorted response instead, and may have no reliable way of knowing that's what happened. An LLM asked to extract a product schema from a block page will often still try, and can return a confident, plausible-looking answer built entirely from a page that was never the actual product. That's fabrication under ambiguity, the same failure mode Section 2.3 describes, just triggered by an access failure instead of a page-content one, and just as invisible to a downstream system with no way to tell the difference.

The compounding risk

Every additional layer between a request and a real answer, network reputation, browser fingerprint, behavioral pattern, challenge response, is one more place a live LLM session can fail silently, on top of the determinism risk Section 2 already describes.

4. Why This Matters Even More for Institutional Users

Every organization that depends on scraped web data cares about accuracy. But investment banks, retailers, and value-added data resellers operate under structural pressures such as regulatory recordkeeping, high-volume automated decisioning, and contractual liability to their own customers, that turn ordinary data-quality risk into a materially different category of business risk.

4.1 Investment banks and asset managers

Job postings, shipping and logistics signals, e-commerce checkout behavior, app-store rankings, satellite and sentiment data, as well as other alternative data derived from the public web have become a mainstream input into equity research and quantitative strategy. That use brings two intertwined obligations that a non-deterministic extraction process cannot satisfy on its own.

The first is recordkeeping and reproducibility. SEC Rule 17a-4 requires broker-dealers to preserve complete and accurate records across communication and data channels, in either a write-once-read-many format or a system with a full audit trail of any modification, retrievable in human-readable form for regulators on request.⁵ A research or trading process that relies on alternative data should be able to show exactly what data was collected, from where, at what time, and by what extraction logic, and to reproduce that same result on demand. A live LLM call that can return a different structured answer for the identical historical page cannot support that kind of reproducibility claim; a versioned, deterministic extraction rule can.

The second is provenance discipline around material non-public information. Compliance guidance on alternative data consistently centers on one requirement above all others: firms must be able to demonstrate that everything collected was genuinely, verifiably public, and was not obtained by circumventing access controls, paywalls, or terms of service that would make it something closer to insider information.⁶ That demonstration is much easier to make, and much easier to defend under audit, when the extraction logic is a fixed, inspectable rule set that a compliance officer can review line by line, rather than an opaque model call whose exact behavior on a given page is not fully knowable in advance even to the team that built it. There is additional value in storing the extraction logic in human-readable form, making it available for audits and reviews.

4.2 Retailers and consumer brands

Retailers and brands run price and assortment monitoring across catalogs that routinely span tens or hundreds of thousands of SKUs and dozens of competitor and marketplace sites, feeding minimum-advertised-price (MAP) enforcement programs and automated repricing engines directly. At that scale, the cost of an extraction error isn't a single wrong data point. It's a wrong input to an automated pricing decision that then propagates to a live storefront, or a false MAP violation flag sent to a brand partner, or a missed one. A repricing algorithm has no way to distinguish a genuine \$49.99 price from a hallucinated one unless the system feeding it guarantees, by construction, that every value came from a fixed and auditable extraction rule rather than a model's best guess on that particular run.

4.3 Value-added data resellers

Value-added data resellers acquire or collect alternative data, apply its own analysis and methodology, and sell the resulting insight to financial institutions and corporations as a research or intelligence product. Firms in this category typically sell structured feeds and research under service-level commitments covering freshness, field-level accuracy, and schema stability: exactly the properties buyers of any data or research product expect to be able to hold their provider to. A single hallucinated field, or a schema that silently varies from delivery to delivery because the extraction step behind it is itself variable, does not stay contained to one bad record: it flows directly into every downstream customer's pipeline, dashboard, or model, and it is the reseller's name and analytical credibility attached to it when something breaks. A

value-added data reseller cannot make a contractual accuracy and consistency commitment on a data product whose underlying extraction step might answer differently each time it looks at the same input.

5. The Ideal Architecture: AI-Assisted Build, Deterministic Run

This whitepaper proposes that a more reliable data pipeline can be built around a simple architectural principle: use an LLM for what it's actually good at, understanding a natural-language request and translating it into a structural plan, and then hand that plan off to a purpose-built, deterministic automation engine for everything that runs in production afterward. The LLM's role ends once a working agent exists. It isn't in the loop on run two, run two hundred, or run two thousand.

5.1 Natural language in, working prototype out

In this paradigm, a user describes, in plain English, what they want an agent to do, including the target sites, the fields to capture, the navigation logic, the output format. The LLM-assisted tool interprets that description and assembles an artifact, the first working version of the agent. This may be in the form of runnable scripts or other reusable stand-alone elements that will perform the work in a reproducible manner, not as a standing prompt that gets re-interpreted on every run.

From this point forward, the agent behaves like any other piece of production software. It can be diffed between versions, rolled back, code-reviewed by a data engineer, and tested against a fixed set of sample pages with a pass/fail result. None of that is possible with a bare LLM prompt, whose “behavior” isn't a static object you can inspect in the first place.

5.2 Additional benefits: human refinement, full control

Once there is a prototype, it can be edited and monitored. A technical user can inspect every step, adjust extraction rules, add conditional logic, handle edge cases the initial prototype missed, and layer in validation: enforcing that a “price” field is always numeric, that a required field is never empty, that a date falls in an expected range. This is the step that turns a fast first draft into a production-grade agent, and it's only possible because the agent is a concrete, editable artifact rather than a black-box model call.

5.3 Deterministic execution at run time

When a finished agent runs, whether on demand or on a schedule, against one page or across a million-row catalog, it executes the same navigation and extraction logic every time, with no model inference involved in interpreting the page. The same page structure produces the same output, run after run. A live LLM call cannot guarantee the same.

5.4 Governance, monitoring, and change management

Because every run executes a fixed, versioned ruleset, a software agent can offer the operational controls institutional data programs actually need:

- Full run-level audit logs: what was fetched, when, using which agent version, and what was extracted. These support the reproducibility and recordkeeping expectations described in Section 4.
- Structural-change detection that flags when a target site's layout has changed and an extraction rule may need attention, rather than silently returning a different, or fabricated, answer.
- Version control and rollback for every agent, so a change can be reviewed, tested, and reverted like any other software deployment.
- Schema and data-quality validation enforced at the platform level, independent of any single run.
- Configurable success criteria that catch problems before they reach production: a run can be checked against what normal looks like, for example flagging a pull that comes back with far fewer records than the same run produced last time, before that shortfall ever reaches a downstream table.
- Automatic re-run when a retrieval comes up short: if an agent, or part of an agent's run, fails to retrieve its expected data, whether that's a single page, a product, or a full day's pull, the platform can detect the gap and rerun it automatically, so a transient failure doesn't turn into a permanent hole in a time series.

The best of both worlds

AI compresses the time it takes to go from a plain-English description to a working prototype from days or weeks to minutes. A deterministic runtime is what lets an institution actually trust, audit, and scale what that prototype becomes.

5.5 Operating reliably against anti-bot defenses

The same principle that applies to extraction applies here: getting past a site's defenses is not a judgment call to make fresh on every run. It's an engineering problem to solve once, then run as configured, versioned infrastructure, the same way the extraction logic itself is. A platform built for production scraping operates this way as a matter of course, rather than asking a model to improvise a new evasion on every attempt:

- Proxy pools with automatic rotation and failover: requests are distributed across large pools of proxies, commonly including residential and mobile IP ranges rather than easily-flagged data-center ranges, with unhealthy or blocked proxies detected and swapped out automatically.

- Real browser rendering: a genuine browser engine, not a bare script, executes a target site's JavaScript challenges the way a real visitor's browser would, rather than presenting the stripped-down fingerprint of a plain HTTP client.
- Configurable pacing: randomized delays between actions and page loads, tuned per site, avoid the uniform request timing that behavioral detection looks for, and double as a way to avoid placing undue load on the source system in the first place.
- Session and cookie continuity: cookies and session state persist across a run the way they would in a real browsing session, instead of resetting with every request in a way no real visitor would.
- Automatic retry and failover: a blocked or failed request triggers a configured retry, often against a different proxy, rather than an ad hoc decision about what to try next.
- Centralized visibility: every run's outcome, including one that got blocked, is visible in one place, so a blocking problem shows up as a specific, addressable event rather than a silent gap in the data.

None of this makes it automatically appropriate to work around a site's access controls. The terms-of-service and site-load diligence described in Section 7.4 still applies regardless of which technical approach is used to reach the page. What changes is that the technique for reaching the page, like the technique for reading it once there, becomes something that can be configured, tested, and audited, instead of something a model has to reinvent, silently and differently, on every run.

6. Illustrative Use Cases

The scenarios below are illustrative composites drawn from patterns common across each industry, not descriptions of any specific customer, engagement, or dataset. They are included to illustrate the preceding argument.

6.1 Investment Bank: Alternative Data for Equity Research

The challenge. A research desk wants a daily hiring-and-demand signal built from job postings across several hundred employer career sites and checkout-availability data from a set of consumer retail sites, to feed a systematic strategy and support published research.

The approach. An analyst describes the target sites and desired fields in plain language to the LLM-assisted agent builder, which produces a working prototype within minutes. The data engineering team then hardens each agent in the editor by adding field validation, restricting sources to a compliance-approved allowlist of genuinely public pages, and configuring the run schedule. Every daily run writes a full audit record of the pages fetched, the agent version used, and the time-stamped, structured output produced.

The outcome. The research desk gets a stable, identically-structured feed every day, with no risk that a numeric field feeding a model was quietly fabricated on a given run. If a regulator or internal compliance

review asks how a specific historical data point was derived, the firm can point to the exact agent version and audit record that produced it. That's a reproducibility standard a live LLM extraction call can't meet.

6.2 Retailer: MAP and Competitive Price Monitoring at Scale

The challenge. A multi-category retailer needs nightly price and availability tracking across roughly 250,000 SKUs on some forty competitor and marketplace sites, both to enforce minimum-advertised-price agreements with brand partners and to feed an automated repricing engine.

The approach. The LLM-assisted agent builder is used to stand up an initial agent for each distinct site template from a plain-language description of the page layout; the merchandising and data teams then adapt each one for category-specific variations across the long tail of product pages. The deterministic runtime executes the full run nightly with schema-validated, unit-normalized output flowing directly into the repricing engine.

The outcome. Run time and infrastructure cost scale predictably with catalog size rather than with per-record model-inference cost. No invented price or availability value reaches the automated repricing decision, and pages whose layout has actually changed are routed to a clear exception queue for a real fix, rather than silently returning a plausible-looking but wrong number.

6.3 Value-Added Data Reseller: Contractual Data Feeds for Enterprise Customers

The category. Analyst-driven research firms use alternative data to build structured feeds and research that they sell to enterprise subscribers under agreements that implicitly or explicitly guarantee high field-level accuracy and a stable, documented schema.

The challenge. A research firm resells structured listing data, for example real-estate or vehicle-marketplace listings, to enterprise subscribers under a service agreement guaranteeing high field-level accuracy and a stable, documented schema.

The approach. The LLM-assisted agent builder is used to rapidly stand up coverage for each new source site the reseller adds. Every resulting agent becomes a versioned artifact that is regression-tested against a library of known-good historical pages before it is promoted into the production feed, and every subsequent run executes that fixed, tested logic.

The outcome. Every subscriber receives an identically structured, field-consistent feed for identical inputs, and the reseller can point to a specific, tested agent version behind any delivered dataset. That combination, a concrete artifact tested once and then run identically thereafter, makes a contractual accuracy and schema-stability commitment defensible. It isn't achievable if the extraction step itself is a live model call that might return a different structured answer for the same page on a different occasion.

7. Risk and Compliance Considerations

The considerations below are general and educational; they are not a substitute for advice from qualified legal and compliance counsel, and specific obligations vary by jurisdiction, regulator, and business model.

7.1 Recordkeeping regimes

Financial-services recordkeeping rules such as SEC Rule 17a-4 require covered firms to preserve complete and accurate records in a form that supports audit and, where applicable, in a write-once-read-many or fully audit-trailed format retrievable on regulatory request.⁵ Any process that feeds research, trading, or client-facing decisions from web-sourced data should be evaluated against this bar, whether or not the firm is itself the covered entity, since value-added data resellers and other intermediaries increasingly find these expectations passed down contractually by their institutional customers.

7.2 Material non-public information exposure

Alternative-data compliance guidance is consistent on this point: firms must be able to demonstrate that scraped information was genuinely and verifiably public, obtained without circumventing logins, paywalls, or terms of service that could render it something closer to non-public or restricted information.⁶ That demonstration depends on the ability to show, precisely, what an extraction process did and did not access. A fixed, reviewable rule set supports that far more directly than a model call whose page-reading behavior isn't fully predictable in advance.

7.3 Data protection and personal information

Where scraped pages may contain personal data, obligations under regimes such as the GDPR require firms to know, precisely and in advance, what fields a process captures, so that personal data can be deliberately excluded or handled under an appropriate legal basis.⁷ A deterministic extraction rule can be scoped to never capture a given field. Asking a model to simply “avoid” capturing a category of information on every run is a request for behavior, not a control, and Section 2 explains why that behavior cannot be relied upon to be identical from one run to the next.

7.4 Terms of service, copyright, and site-load diligence

None of the architectural points in this paper substitute for the underlying legal diligence every organization doing web data collection should perform: reviewing target sites' terms of service, understanding applicable copyright and database-right exposure, and ensuring collection does not impose undue load on source systems. This diligence should also cover a site's stated policy on automated access itself. Some sites' terms explicitly prohibit circumventing bot detection, and the technical ability to get past a block, discussed in Section 3, is not the same thing as having the right to. What a deterministic platform

adds is that once that diligence is done for a given source, the resulting extraction rule enforces the same scope and behavior on every run, rather than requiring that diligence to somehow be re-verified against a model's judgment call each time it happens to read that page.

8. Conclusion and Recommendations

Large language models have changed how quickly a web-scraping capability can be built. They haven't changed the fact that production data operations require deterministic, auditable, versioned behavior. Institutions that route ongoing extraction directly through a live LLM call are, whether they intend it or not, accepting a level of run-to-run variability that is difficult to detect, difficult to explain to an auditor or a customer, and inconsistent with the recordkeeping and provenance obligations that apply to investment banks, retailers, and value-added data resellers alike.

The model proposed in this whitepaper is built on the alternative: let an LLM turn a plain-language description into a working first draft, then refine and harden it through editing and validation, and run it as a concrete, versioned, testable piece of software, forever after. That is how an organization gets the speed institutional teams want from AI without giving up the reliability institutional data programs require.

Recommendations

- Use an LLM as a developer of your extraction logic, not as the runtime that executes it in production.
- Require that any production scraping process can reproduce a specific historical result on demand, and can show the exact rule version that produced it.
- Build compliance and schema validation into the extraction platform itself, rather than relying on a model to apply the same judgment call identically on every run.
- Evaluate AI-assisted scraping tools on what happens after the prototype is built, not only on how quickly the first version comes together.
- Know your data, its provenance, and its extraction logic. Require full transparency into where data originates, what was collected, how it was transformed, and which extraction logic produced it.
- Avoid black-box data and production architectures in which critical outputs depend on opaque, non-repeatable model behavior.

Author's Note: BC Wilson is Chief Operating Officer at Sequentum. The principles in this whitepaper have guided Sequentum's approach to web data extraction for nearly two decades. If you're working through these questions, we'd welcome the conversation. Reach out at info@sequentum.com or visit sequentum.com/cloud. This whitepaper was written with assistance from Claude.

References

- [1] Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang, “An Empirical Study of the Non-determinism of ChatGPT in Code Generation,” [An Empirical Study of the Non-determinism of ChatGPT in Code Generation](#).
- [2] On the hardware- and inference-level sources of non-determinism and why temperature and seed controls do not fully eliminate it: [“Understanding and Mitigating Numerical Sources of Nondeterminism in LLM Inference,” arXiv:2506.09501](#); [“Necessary but Not Sufficient: Temperature Control and Reproducibility in LLM-as-Judge Safety Evaluations,” arXiv:2606.26185](#); [“Why Temperature=0 Doesn't Guarantee Determinism in LLMs”](#); Abstract, [“Why Is Deterministic Output From LLMs Nearly Impossible?”](#).
- [3] On adaptive, non-deterministic-by-design agent harnesses for web automation: [OpenClaw Browser Harness documentation](#).
- [4] [“On the Internet, Nobody Knows You're an LLM Bot: Unmasking Web Agents with Multi-Layer Fingerprinting,” arXiv:2606.30119](#).
- [5] U.S. Securities and Exchange Commission, [“Amendments to Electronic Recordkeeping Requirements for Broker-Dealers”](#); see also [Smash, “SEC Rule 17a-4: Electronic Recordkeeping Requirements Explained”](#).
- [6] U.S. Securities and Exchange Commission, Division of Examinations, [“Investment Adviser MNPI Compliance Issues” Risk Alert \(April 26, 2022\)](#).
- [7] [CNIL, “The Legal Basis of Legitimate Interest: Focus Sheet on the Measures to Implement in the Case of Data Collection by Web Scraping”](#); see also [European Data Protection Board, “EDPB Sheds Light on Anonymisation and Web Scraping for Generative AI and Adopts Final Version of Guidelines”](#).