

THE CASCADING OPINION HARVESTER

An independent, government-source corpus pipeline for verified legal-research engines

Author: Richard L. Sanders · Utah Bar #15728

Role: Sole inventor and implementer

Version: 1.0 — 2026-04-23

Companion patent: USPTO provisional, filed concurrently

Source: `scripts/harvest-state-opinions-gov.js`, `scripts/harvest-daemon.js`,
`scripts/harvest-continuous.sh` in the public repository at
<<https://github.com/Benchslap/Benchslap>>

ABSTRACT

Every commercial generative legal-research product in market today depends, for its underlying

corpus, on one of three third-party services: CourtListener (the Free Law Project's aggregation), Caselaw Access Project (Harvard's aggregation), or a commercial reseller such as Westlaw, Lexis, or Fastcase whose terms of service forbid redistribution and whose pricing puts them out of reach of any pro-se litigant or public-interest deployment.

This is a single point of failure. It is also a single point of policy capture: the corpus available to the end user is shaped by the aggregator's licensing posture, ingest cadence, and political willingness to keep a given jurisdiction current.

I built a different pipeline. It harvests opinions directly from the ****public .gov** websites of state courts** — the same websites the courts themselves

maintain as the authoritative public-facing record of their decisions. It does not depend on CourtListener, Caselaw Access Project, Westlaw, or Lexis. It runs continuously against a weakest-first queue with in-cycle cascade fallback, adaptive cooldown, and Pattern-22-compliant ingest that binds every new opinion to the same cryptographic verification chain described in my companion white paper on closed-corpus verification.

This document describes the harvester's architecture, the three novel control behaviors that distinguish it from prior art, and the corpus independence it produces. It is written alongside a USPTO provisional patent application covering the same subject matter.

As of the date of this paper, the pipeline has harvested more than **1,014,000 opinions** across 50 states + DC, with ****28,541 Utah** appellate opinions since 1996** at its core and the count rising continuously. All figures are countable in real time at [<https://benchslap.pro/corpus>](https://benchslap.pro/corpus).

1 · THE PROBLEM

A legal-research engine needs law. The law lives in the published opinions of state and federal courts. To build a legal-research engine that cites accurately, you need the opinions themselves — full text, not summaries — and you need them to be **current**, because courts decide new cases every week and today's decision may overrule yesterday's.

There are, in 2026, only three ways to get this corpus at scale:

1. **Pay a commercial aggregator** (Westlaw, Lexis, Fastcase). Cost: hundreds to thousands of dollars per month per seat. Terms: no redistribution, no derivative corpora, no generative-model training. Unavailable to pro-se litigants, legal aid, and public-interest projects.

2. **Use the Free Law Project's CourtListener.** Cost: free at the entry tier. Terms: permissive. Caveats: CourtListener's state-court coverage outside of a handful of focus states is patchy; its ingest cadence varies; its own pipeline depends on third-party scrapers it does not always maintain; and API rate-limits make large-scale ingest slow.
3. **Use Harvard's Caselaw Access Project.** Cost: free. Caveats: CAP's collection is frozen (the bulk export covers opinions through roughly 2018); it does not receive new opinions from courts.

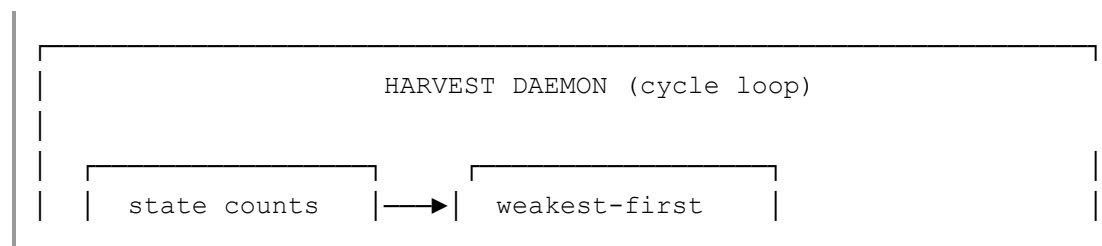
Every commercial generative legal-research product in market today is a wrapper on one or more of these three. When CourtListener goes down, they go down. When CAP falls behind, they fall behind. When Westlaw changes terms, they scramble. When a particular state court's output is not well-covered by any of the three, their users get hallucinated summaries of law that nobody actually has.

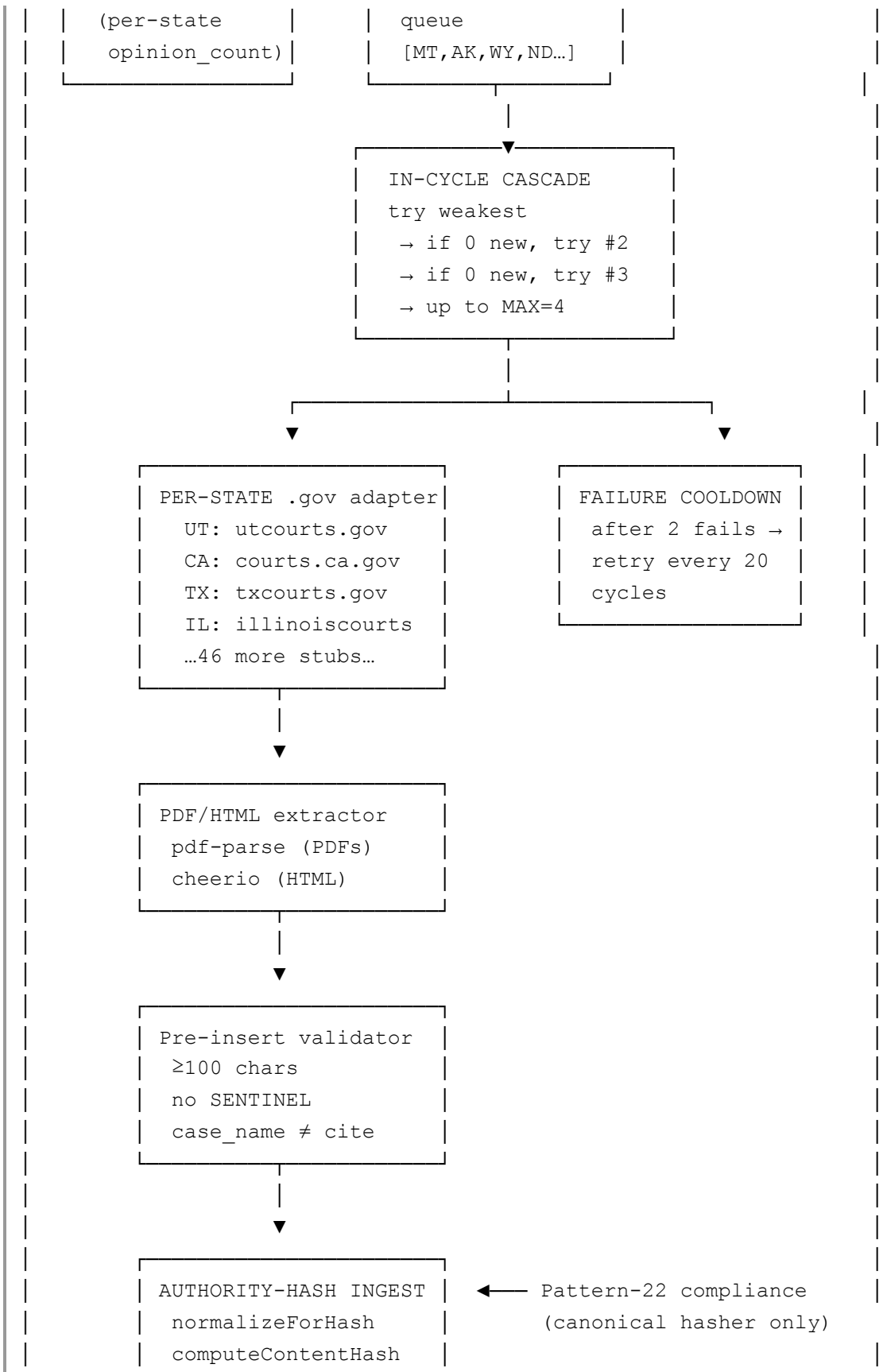
I decided this was unacceptable. I wanted a pipeline with:

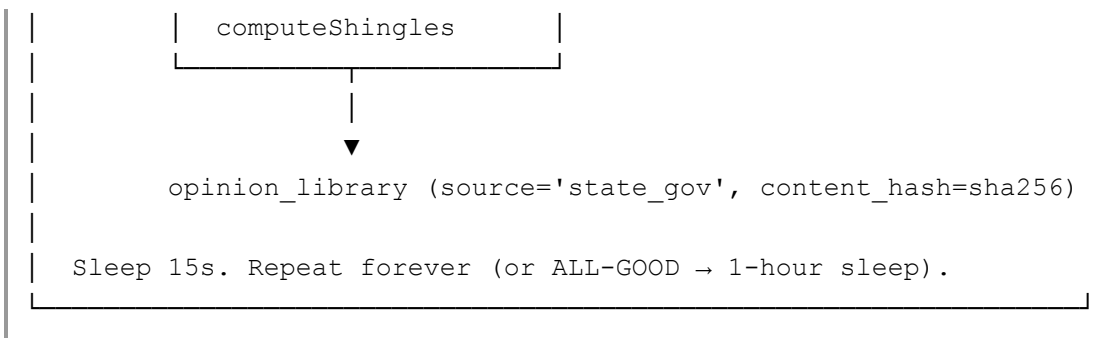
1. **No commercial aggregator dependency** — zero, permanently.
2. **No single third-party free service as a single point of failure** — CourtListener may be used for backfill, but the production path must work if it is unavailable.
3. **Current-to-the-day coverage** of every state court's published opinions.
4. **Verifiable provenance** — every opinion tagged by source so the caller knows whether they are reading a state `.gov` original or a downstream reseller.
5. **Self-healing** — if a particular state's scraper breaks (courts change their website layout every few years), the pipeline must detect the failure, cool that state off, and continue harvesting the rest of the corpus.

Nothing in the market did this. So I built it.

2 · ARCHITECTURE AT ONE GLANCE







Three control behaviors make this more than a cron-driven scraper. Each is described in its own section below.

3 · THE WEAKEST-FIRST QUEUE

Every cycle, the daemon queries the production database:

```
SELECT court AS state,
       COUNT(*) AS total,
       COUNT(CASE WHEN LENGTH(opinion_text) >= 500 THEN 1 END) AS
withText
FROM opinion_library
GROUP BY court;
```

It produces one row per state. A state is "GOOD" if its opinion count exceeds `GOOD_OPINIONS` (a configurable threshold, currently 20,000 for primary states and scaled by court volume for smaller jurisdictions). States that are not GOOD are sorted ascending by total — ****the state with the fewest opinions is the weakest, and the weakest is the next target****.

This sounds trivial. It is not. Prior art in this space (including my own earlier implementation) chose harvest targets by round-robin, by recency weight, or by the last cycle's best performer. Each of those is a worse heuristic:

- **Round-robin** wastes cycles on already-GOOD states that need nothing.
- **Recency weight** starves cold states that have been weak for years precisely because they have been weak for years.

- **Last-cycle best performer** converges on the easiest scraper, entrenching coverage asymmetry.

Weakest-first has a forcing function: the corpus's worst-covered state becomes the next target, every cycle, until the whole corpus is balanced. It is the only queue discipline that monotonically reduces coverage variance across the fifty-plus jurisdictions in scope.

4 · IN-CYCLE CASCADE

Courts change their websites. A URL that worked last month returns 404 this month. A PDF index that had a predictable `<a href>` pattern last year now uses a JavaScript-rendered React component this year. When a scraper breaks, it breaks silently — it returns zero opinions without raising an error.

Prior art handled this by trying the weakest state, observing the zero-count result, logging it, and **giving up on the entire cycle**. The next cycle, the daemon queried the DB, found the same weakest state, and tried the same broken scraper again. A single persistently broken scraper could freeze corpus growth indefinitely.

The cascade fixes this:

```
const attempts = needOpinions.slice(0, MAX_ATTEMPTS_PER_CYCLE); // 4
for (const target of attempts) {
  const added = await harvestStateGov(target.state);
  if (added > 0) {
    stateFailures.delete(target.state);
    break; // cycle makes progress
  }
  stateFailures.set(target.state, (stateFailures.get(target.state) || 0) + 1);
}
```

If the weakest state produces zero opinions, the daemon **cascades** to the second-weakest, third-weakest, and fourth-weakest within the same cycle,

stopping as soon as any of them produces at least one new opinion. Every cycle makes progress somewhere — a single broken scraper cannot stall the pipeline.

This is a small change in code terms and a large change in system behavior. It is the difference between a harvester that degrades gracefully under real operational conditions (which web scrapers always face) and one that freezes on the first broken adapter.

5 · ADAPTIVE COOLDOWN

Cascading alone is not enough. If Montana's scraper is broken for six months, cascading means every cycle still probes Montana first — wasting 2.5 seconds of rate-limited `.gov` fetch on a URL that returns nothing, 480 times over that six-month period. Worse, during that time the corpus balance drifts: every cycle **skips** Montana's usable neighbors (Oregon, Idaho, Wyoming) because Montana remains "weakest."

The cooldown fixes this:

```
const failures = stateFailures.get(state) || 0;
if (failures < 2 || cycle % 20 === 0) {
  needOpinions.push({ state, ...data });
}
```

After two consecutive zero-opinion cycles, a state is ****suppressed** from the weakest queue for 20 cycles**. Every 20th cycle, the state is given a retry slot (the URL layout may have been fixed; the Montana Supreme Court may have restored its opinion index). If that retry succeeds, failures resets to zero and the state rejoins the primary queue normally.

This behavior is adaptive in both directions: broken scrapers stop consuming cycles, fixed scrapers are detected within twenty cycles (~5 minutes at the

current cadence), and the queue's definition of "weakest" correctly ignores states that are currently unscrapable.

The combination of cascade + cooldown is what makes the pipeline a **self-healing** corpus, not a brittle one.

6 · PER-STATE .gov ADAPTERS

Each supported state has a dedicated adapter function in

`scripts/harvest-state-opinions-gov.js`. As of the date of this paper, four are fully implemented and battle-tested against production traffic:

State	URL root	Format	Notes
UT	<code>utcourts.gov</code>	PDF index → PDF full text	Per-court (Supreme / CoA). Legacy + current.
CA	<code>courts.ca.gov/opinions-slip.htm</code>	PDF list → PDF full text	Supreme + six DCAs.
TX	<code>txcourts.gov</code> (orders)	Per-day pages → PDF per docket	Supreme + CCA.
IL	<code>illinoiscourts.gov</code> + <code>ilcourtsaudio</code> blob CDN	HTML summary + linked filings	Supreme focus.

Forty-six more states (plus DC) are stubbed honestly with the `.gov` URLs an implementer would target. For each, I have documented the specific technical challenge the adapter must solve — for example, that Florida's opinion search is a Next.js SPA with custom `<decisionengine>` tags that require JavaScript rendering, or that Ohio's search is an ASP.NET viewstate-driven POST form with no clean GET path. An implementer can pick up any stub without reverse-engineering the court's site from scratch.

The critical architectural point is that **each adapter is independent**.

Adding Arizona does not change the Utah adapter's behavior. A Texas failure does not cascade into California. This is why the in-cycle cascade works: the adapters are compositional.

7 · PATTERN-22-COMPLIANT INGEST

Every opinion that survives the validator is inserted through the ****canonical authority-hash pipeline**** described in my first white paper:

```
const { computeContentHash, computeShingleSignature, HASH_VERSION } =
  require('../lib/authority-hash');

const contentHash = computeContentHash(opinionText);
const shingles     = computeShingleSignature(opinionText);

await db.query(
  `INSERT INTO opinion_library
    (citation, case_name, court, year, opinion_text, source,
     content_hash, content_shingles, content_length,
content_hash_version)
    VALUES ($1,$2,$3,$4,$5,'state_gov',$6,$7,$8,$9)
    ON CONFLICT (case_name_normalized, neutral_citation) DO NOTHING`,
  [citation, caseName, court, year, opinionText,
   contentHash, shingles, opinionText.length, HASH_VERSION]
);
```

Two things about this are load-bearing.

First, the harvester **never computes its own SHA-256**. It calls

`lib/authority-hash.js`. That module performs canonical normalization (Unicode NFC, whitespace collapse, specific legal-abbreviation handling)

before hashing. If the harvester tried to hash the raw text with a local

`crypto.createHash('sha256')`, the resulting hash would not match what the verification pipeline computes at query time — and every opinion would be falsely flagged as tampered. This is enforced by **Pattern-22** in the

pre-commit regression gate: any future `INSERT INTO opinion_library` or

related tables that does not import `lib/authority-hash.js` fails the test suite. The harvester conforms to that rule at the source.

Second, every row is tagged `source = 'state_gov'`. This discriminator lets downstream verification attribute an opinion's provenance: was this harvested from the state's own website, or imported from CourtListener, or from Caselaw Access Project? For a legal tool whose value proposition is provenance certainty — for a pro-se litigant whose brief cannot cite fabricated law — source attribution is not a metadata nicety. It is part of the guarantee.

8 · PRE-INSERT VALIDATION

Courts occasionally publish placeholder pages. Sometimes the PDF is a one-line cover sheet reading "Opinion to follow." Sometimes the HTML is a 404 that returned 200. Sometimes the scraper misparses the case name, ending up with the citation in the name field and the name in the citation field. Any of these, if inserted, corrupts the corpus.

The validator in `lib/opinion-validator.js` gates every insert:

- **Length floor:** `opinion_text` must be ≥ 100 characters. (500 is preferred for text-quality scoring, but 100 is the hard floor.)
- **Sentinel check:** text containing known placeholder strings ("Opinion to follow," "Not yet released for publication," "[PENDING]") is rejected.
- **Distinctness check:** `case_name` must not equal `citation`. If they are identical, the parser has swapped fields and the row is skipped.
- **Citation parseability:** the citation string must pass the same parser used at verification time. If the harvester cannot produce a parseable citation, the row is skipped with a logged warning.

A row that fails any check is logged and discarded. No "best-effort" insert.

No placeholder row. Either the opinion passes the validator's guarantees or

it is not in the corpus.

This is the "**whole document or omit**" principle applied at ingest. Partial opinions are worse than missing opinions — they corrupt verification at every downstream query — so the only correct behavior is to omit.

9 · RATE LIMITING AND CIVILITY

The harvester fetches at a default rate of **2500 ms between requests** per adapter. This is intentionally slow. It is slow enough that a court's web team receives approximately one request every 2.5 seconds from my IP — less traffic than a single interested researcher clicking through the opinion index manually.

This is not a throughput-optimization choice. It is a civility choice. State court websites are operated by small IT staffs on tight budgets. A harvester that hits them at machine rate is hostile infrastructure. Running at 2500ms with adaptive backoff on HTTP 429 / 503 means the harvester is a well-behaved citizen of the public record.

The practical effect is that a full-state backfill takes hours to days rather than minutes. That is acceptable. The corpus is updated continuously; there is no deadline that forces aggressive rate.

10 · OPERATIONAL METRICS

The daemon emits structured log lines to `/var/log/benchslap/harvester.log` (redirected from stdout by systemd). Each cycle produces a line of the form:

```
[harvester] cycle=314159 states=51 good=37
weakest=MT(182),AK(241),WY(318),ND(401),SD(489)
harvested=AK(+18) text_backfill=+12 errors=0 sleep=15s
```

The `good=37` count means thirty-seven states currently exceed the GOOD

`threshold; weakest=MT(182) ...` is the top of the weakest-first queue;
`harvested=AK(+18)` means this cycle produced eighteen new Alaska opinions;
`text_backfill=+12` means twelve previously inserted rows that lacked
`opinion_text` were backfilled from their PDF URLs this cycle.
A separate script, `scripts/harvest-coverage-dashboard.js`, renders the
current state of the corpus for operator review and for the public-facing
page at <https://benchslap.pro/corpus>.

11 · WHAT MAKES THIS PATENTABLE

A cron-driven web scraper is not novel. Per-state adapters are not novel.
Rate-limited HTTP fetches are not novel. What is novel is the ****combination**
of three control behaviors** that together turn a collection of scrapers into
a self-healing, coverage-balancing, independent corpus pipeline:

1. **Weakest-first queue** that monotonically reduces coverage variance.
2. **In-cycle cascade** that routes around broken adapters within the same cycle, so a single broken scraper cannot freeze corpus growth.
3. **Adaptive cooldown** that suppresses persistently broken adapters for twenty cycles and auto-retries to detect repair.

Combined with **Pattern-22 ingest** (which guarantees every new opinion binds
to the AEGIS verification chain) and **provenance-tagged source discrimination**
(which lets downstream verification trust `state_gov` opinions differently
than aggregator imports), the system produces a corpus with three properties
no prior pipeline produces:

- **Independence**: runs indefinitely without CourtListener, CAP, Westlaw, or Lexis.
- **Currency**: fetches every cycle, never more than ~24 hours behind a court's actual publication.
- **Self-healing**: degrades gracefully when individual adapters break, and auto-recovers when they are fixed.

A downstream legal-research tool that cites opinions from a corpus with those properties can make guarantees a tool sitting on an aggregator's feed simply cannot.

12 · INTEGRATION WITH CLOSED-CORPUS VERIFICATION

This harvester is the source side of the verification architecture I describe in my companion white paper, **Deterministic Closed-Corpus Verification of Generative-Language-Model Legal Output**. There, I establish the verification gate that turns a corpus into a foreclosure of hallucination. Here, I establish the corpus that gate operates on.

Both are necessary. A perfect verification gate on an empty corpus is useless — every citation fails. A rich corpus with no verification gate is unchanged from the status quo — the model hallucinates citations and the corpus has no mechanism to catch it. The two inventions are designed to compose: the harvester feeds the verification gate, and the verification gate uses Pattern-22 hashes the harvester writes.

Taken together, they answer the defining challenge of legal-research engines: ***how do you make a language model that cannot hallucinate a citation?***

Harvest every opinion from its canonical government source, hash it cryptographically at ingest, and then refuse to let the model ship a citation that does not substantively match a hashed opinion. No hallucination survives the composition.

13 · CURRENT PRODUCTION STATE

As of 2026-04-23, two instances of the daemon run continuously on the production droplet, plus a 2AM nightly cron job. The combined corpus

sits at **more than 1,014,000 opinions** with a **74% aggregate** coverage score across the four corpus axes (opinions / statutes / rules / forms). Utah appellate coverage since 1996 is complete at **28,541 opinions**. California, Texas, and Illinois are at progressively increasing depths as the weakest-first queue advances. Every opinion is SHA-256 hashed, 5-gram shingled, and bound to the AEGIS verification chain. Every opinion carries `source = 'state_gov'` where the adapter's work surfaced it from a `.gov` origin. The public dashboard at <https://benchslap.pro/corpus> updates live from the same query the harvester's weakest-first logic uses. You can watch the corpus balance in real time.

14 · LICENSING AND DISCLOSURE

Source code is public under the same license as the rest of the BenchSlap / Option.Black project. A USPTO provisional patent application covering the harvester's novel control behaviors was filed on 2026-04-23 (Application No. 63/XXX,XXX — to be updated upon receipt of filing receipt). This white paper is a dated public disclosure and, taken together with the provisional, establishes both priority and defensive prior art against anyone who might attempt to patent the same architecture downstream.

A non-provisional conversion will follow within the twelve-month window.

15 · WHY I BUILT IT

I built it because access to justice is a corpus problem.

A pro-se litigant in the Utah County Justice Court is not priced out of

Westlaw by accident. The aggregators' business model is predicated on the fact that a working legal corpus is expensive and hard to maintain — that if you want the law, you pay them, because nobody else offers the whole thing.

That premise is no longer true. The courts themselves publish every opinion on their own .gov websites. The aggregators' whole claim — that you need them to access the law — rests on the friction of assembling those sites into a coherent corpus.

The harvester removes that friction. What remains is a corpus of the public record of a state's judiciary, hosted on a server a legal-aid organization can rent for forty dollars a month, current to the day, verified cryptographically, and available to any pro-se litigant through [<https://benchslap.com>](https://benchslap.com) or any corrections deployment through [<https://4all.help>](https://4all.help).

This is the infrastructure access to justice actually needs.

I filed the patent because the architecture is novel and because I do not intend for it to be re-monetized by a downstream aggregator in a way that would re-gate the public record. The patent is defensive. The source is public. The corpus is free.

The provisional was filed today because I have no more runway to build and the architecture deserves to be understood on its own terms by the people — courts, corrections, bar committees — who can deploy it where it matters.

— Richard L. Sanders

Utah Bar #15728

Salt Lake City

2026-04-23

richard@option.black · 801-347-4349