Career Ops

A local, event-sourced pipeline that discovers roles, scores them against a written profile, derives application state from an email corpus, and renders a single-file dashboard. Python and SQLite, no hosted services, no third party holding the data.

Technical summary · Benjamin J. Sunter · github.com/bsunter93/careerops · MIT

1 · PROBLEM AND THESIS

The average open role now draws more than 300 applications, up from roughly 100 in 2021. AI made applying nearly free; the number of humans who can read applications did not change. Volume is what broke the channel, so the tool deliberately does the opposite of what most job-search software does: it never writes or submits anything. It reduces how many roles are worth attention, and it keeps an honest record of what actually happened to the ones that were.

Two claims drive every design decision downstream.

> **Fit is a gate, not a dial.** You clear the bar for a role or you do not. Ranking a 145-day-old 85 above a 3-day-old 78 is actively misleading.

> **Among roles you clear, posting age dominates.** A hiring manager on a desirable remote req described screening roughly the first 200 of 6,000 applicants and stopping, and called that normal. Being early is a large advantage unrelated to the résumé. So candidates are **banded by age first and ranked by fit inside the band**.

2 · SCOPE

5,194	768	536	500
LINES, 17 MODULES	EVENTS INGESTED	APPLICATIONS	ROLES / 159 COMPANIES

Subsystems

- **discover** — poll public ATS boards, normalise postings
- **aggregators** — broad-reach feeds and ATS slug discovery
- **gmail / ingest** — fetch and store the mail corpus
- **classify** — 609 lines of event typing
- **resolve** — 724 lines of identity resolution
- **fit** — LLM scoring against a written profile
- **intel** — public employee sentiment, reference only
- **resume** — deterministic bullet selection, Word render
- **dashboard** — 1,209 lines, self-contained HTML projection
- **db / schema** — 8 tables, 1 view, derived status

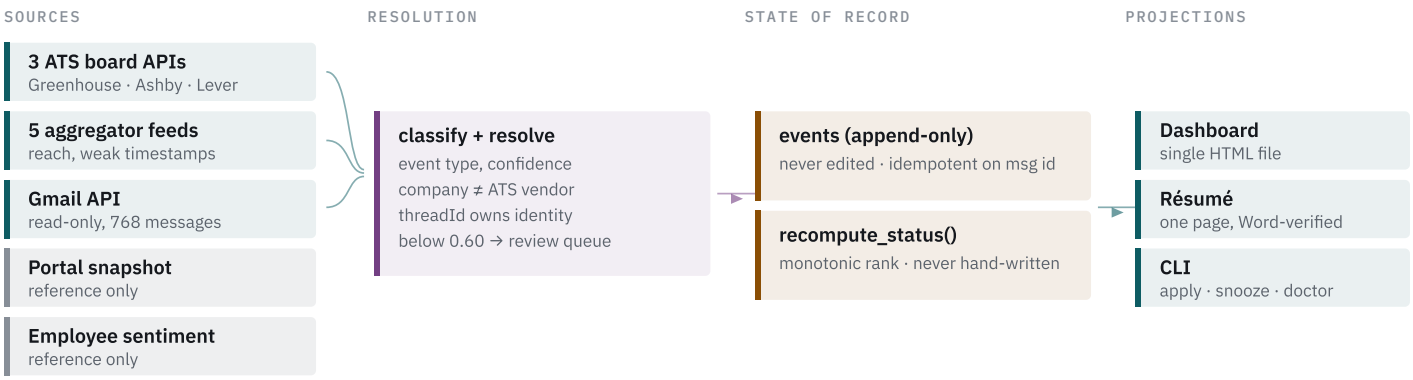
Corpus today

Submitted applications	343
Open prospects	193
Events retaining body text	721 / 768
Roles with publisher timestamp	201
Roles scored	213
Companies with sentiment	14
Unresolved, queued for a human	7

3 · CORE MODEL

Applications are entities with derived state. Emails are events that mutate it. Everything follows from that one decision.

The predecessor was a spreadsheet with one row per email, which is why it held sender addresses in a "Company" column. One row per *application*, with an append-only event log beside it, is the whole fix.



Projections are disposable and regenerable. Nothing downstream of the event log is ever the record.

Reference sources (grey) never write derived status. A third-party review aggregate or a portal screen must not quietly promote an application.

Schema

`companies → roles → applications → events`, plus `artifacts`, `review_queue`, and two reference tables. Status is derived by `recompute_status()` from the event log and is never written by hand; `STATUS_RANK` makes it monotonic, so a late acknowledgement cannot downgrade an interview. `prospect` ranks below `applied` at −1: without that, a discovered role silently became "applied" and the tool claimed submissions that never happened.

4 · DATA SOURCING AND TRANSLATION

SOURCE	GIVES	KNOWN WEAKNESS
Greenhouse / Ashby / Lever board APIs	Full JD text, comp when published, and the publisher timestamp (`first_published`, `publishedAt`, `createdAt`)	Only covers employers on a known slug; 87-entry watchlist
5 aggregator feeds	Breadth beyond the watchlist, plus ATS slug discovery	Weak or absent posting dates, which is the variable that matters most
Gmail API (read-only)	The entire outcome record: acks, rejections, invitations	The search query is the one irreversible gate in the pipeline
Employer portal snapshot	Role identity for acknowledgements that name no role	Wrong in both directions; never feeds status
Public employee sentiment	Rating, sub-ratings, pros and cons, with sources	Self-selected reviews; same-name ambiguity; never feeds fit

No LinkedIn or Indeed scraping: against their terms, brittle, and unnecessary given public board APIs. Sentiment is gathered through a server-side web search reading public result snippets, which is what a person gets from a search engine, rather than scraping a review site behind bot detection and a work-email gate.

Translation is the hard part

Turning a mail corpus into an application ledger is almost entirely a resolution problem, which is why `resolve.py` and `classify.py` are the two largest modules in the system.

- **ATS vendors are not employers.** Domain *suffix* matching, so `us.greenhouse-mail.io` never becomes a company.
- **The From display name is the best company signal.** `Match Group <no-reply@hire.lever.co>` carries the employer where the domain carries only the vendor.
- **Most acknowledgements name the role only in the body**, so `role_from_body()` carries roughly fourteen patterns because every ATS phrases it differently.

- **Gmail's threadId owns application identity.** Later messages in one conversation resolve slightly different titles, so "Launch PgM" and "Launch Program Manager" became separate applications and one interview loop counted as several advances.
- **Thread alone is insufficient in both directions.** Gmail also threads on identical subjects, so two real applications sharing one subject line must not be folded; `merge_threads` clusters by title compatibility inside a thread.
- **Never guess.** Below 0.60 confidence, or missing company or role, goes to the review queue. No JD means no fit score: `fit.py` returns `None` rather than a number, because bad data is worse than absent data.

5 · REFRESH MECHANICS AND CADENCE

One daily job, because posting age is the dominant variable and the point is to see a requisition while it is days old rather than weeks.

STEP	DOES	REPLAYABLE
`sync`	Fetch new mail, classify, attach to applications	Yes, idempotent on `gmail:<msgid>`
`discover`	Poll boards, normalise, record publisher timestamps	Yes, unique on company + title
`fit --limit 25`	Score the backlog against the written profile	Yes, but metered for API cost
`dashboard`	Re-render the single-file projection	Yes, pure projection

> **Ingestion is idempotent; the search query is not.** Everything downstream of storage can be re-run against stored events. A message never fetched cannot be reclassified later, so under-fetching is the one irreversible failure mode, and over-fetching costs only a trip through the classifier.

Events retain their body (2,000 characters). A system that cannot reproduce its own classifications cannot be corrected safely, and rejection language lives in the body rather than the subject. `reclassify` is a no-op when nothing has changed, and touches only `source='gmail'` so it cannot re-judge a manually recorded submission from an empty subject line.

Pacing is measured on rolling 7-day windows, never calendar weeks. The current calendar week is partial six days out of seven, so comparing it to a finished week always reports a collapse in output that did not happen.

6 · RANKING AND DECISION LOGIC

- **Age bands first, fit inside the band.** Any ranking that puts a stale high score above a fresh qualifying one is misleading.

- **Scoring covers capability and preference.** Dealbreakers in the profile cap the score under 40 regardless of skill match; encoding an IC-track preference moved one role from 79 to 22 because its mandate was building a team.
- **Exclusions are domain and level, never job family.** Excluding "technical program manager" wholesale blocked monetisation TPM roles scoring 72, squarely in the target background. The real exclusions are infrastructure, security, compute, machine learning.
- **The comp floor never filters on absence.** Most JDs publish no range, so only a *parsed* maximum below the floor disqualifies.
- **Dormancy is per company and non-destructive.** A default of 21 days, overridden where an employer states a longer review window; a later event revives the application automatically. Stale rows are never deleted, because deletion loses the event history.
- **Snooze is the primitive.** Employer application caps are undocumented and change, so a cap is simply a computed reason to hide something, not a hardcoded rule.

7 · PRINCIPAL CORRECTIONS

The failure mode throughout was the same: software stating something untrue with total confidence, and being believed. Most of the engineering effort went into teaching it to say when it does not know.

DEFECT	CAUSE	IMPACT
Discovered roles reported as submitted	`prospect` outranked `applied` in status derivation	49 records
Outcomes never recorded at all	Mail failed both arms of the search query, so it was never fetched	44 events 13 applications
Genuine rejections destroyed on reclassify	Bodies were not retained, and rejection language is not in the subject	48 rejections
Stored bodies held CSS, not text	Tags stripped before `<style>` blocks; one vendor's font block filled the buffer	81 events
Acknowledgements read as rejections	"If you are not selected…" matched inside a hypothetical clause	14 acks
False interviews	Weak signals ("next steps") matched boilerplate in the body	22 → 7 real
Advance rate inflated	The definitive-ack guard covered only two of the promotable event types	3% → 5% overstated
One interview loop counted as several	Title variants inside one thread created separate applications	6 vs 2 true
Closed and filled requisitions ignored	Closure language was not treated as a rejection; one read as forward progress	12 events 6 unresolved
Applications silently mislabelled	Renaming a shared role row rewrote the title for every application on it	12 applications
Backfill was a silent no-op	The refetch predicate was not widened when a stored field was added	686 events
Phantom applications	Any mail with a parsable sender domain created an application row	gated on event type
Contract body-shops read as interviews	They use real interview language about roles never applied for	demoted on body tells

Two structural lessons came out of these. **First, a pipeline needs a quarantine, not a default.** Seven items currently sit in the review queue rather than being guessed at. **Second, the most destructive misread available is closing a live thread**: the role leaves the decision list and the record says it is dead, so it is never followed up. Errors in that direction are worth spending disproportionate effort to prevent.

8 · DELIBERATE LIMITS

- **It does not fix the problem.** Nothing on the candidate side can; the bottleneck is how many humans can read. It reduces attention spent on roles that were never going to answer.
- **It writes and submits nothing.** Applying happens in a browser; `careerops apply <id>` records the submission as an event so status still derives from the log.
- **Reference data never feeds ranking.** Portal status and employee sentiment are displayed with provenance and confidence, and are excluded from fit and from derived status by construction.
- **The dashboard is a projection, never a record.** It is regenerated from SQLite each run; hand-editing it would create a second competing source of truth.
- **Coverage is bounded by the watchlist.** 87 employer board slugs plus aggregator reach. Roles with no publisher timestamp are usually delisted, which is itself a signal.

Local Python 3 and SQLite in WAL mode. Gmail access is read-only under `gmail.readonly`; the OAuth token stays on disk and no message content is transmitted anywhere except to the scoring model for roles explicitly queued. Fit scoring and sentiment run on the Anthropic Messages API. MIT licensed.