Agnostos

Resume-driven academic outreach for research internships

Architecture, algorithms, scoring and design decisions

Avinash Chandra Gupta

https://github.com/Avinashistoocool/Agnostos-v2

Version 2.0.0 October 2026

Abstract

Agnostos is an open-source command-line tool that helps a student find research internships. It reads the student's resume, builds or reads a list of professors, ranks that list locally against the resume, pulls each professor's recent first- and last-author papers from OpenAlex, asks a language model for one acceptance estimate and one short email per professor, strips the usual machine-written tells out of the draft, checks and repairs the email addresses, sends the mail slowly from the student's own account inside each professor's working hours, and then tracks replies and drafts follow-ups. The spreadsheet is the only database. This report documents the motivation, the architecture and every scoring rule in the code: the TF-IDF matcher and its saturating term bonus, the topic filter, the OpenAlex discovery ranking with its relevance floor, field anchoring, seniority gate and prestige bonus, the batch planner, the publication selection used for enrichment, the token and cost estimator, capability negotiation with OpenAI-compatible gateways, the AI-tell score, the address confidence model and the web finder's evidence score, the professor recheck, the send-window and meeting-slot schedulers, reply classification, and the safeguards that keep the tool from mailing someone twice or promising something the student cannot deliver.

Contents

1 Motivation **3**
- 1.1 Design principles . . . 3

2 System architecture **3**
- 2.1 Components . . . 4
- 2.2 The pipeline in six steps . . . 5

3 Resume parsing **5**

4 Target acquisition **5**
- 4.1 Mode B: an existing workbook . . . 5
- 4.2 Mode A: directory scraping . . . 6
- 4.3 Topic focus . . . 6

5 Discovery from a research topic **6**
- 5.1 Which papers count . . . 6
- 5.2 Crediting authors . . . 7
- 5.3 Ranking . . . 7
- 5.4 Batches . . . 7

6 OpenAlex enrichment **8**

7 The local matcher **8**
7.1 Text normalisation and features . . . 8
7.2 TF-IDF cosine . . . 8
7.3 Saturating domain-term bonus . . . 8
7.4 Combined score and selection . . . 9

8 Language-model evaluation and drafting **9**
8.1 One call per professor . . . 9
8.2 Token and cost estimate . . . 9
8.3 Capability negotiation . . . 10

9 The humanising layer **10**
9.1 Stage 1: deterministic deslop . . . 10
9.2 Stage 2: the AI-tell score . . . 10
9.3 Stage 3: conditional rewrite . . . 11
9.4 Self-funding guard . . . 11

10 Email verification and repair **11**
10.1 Local confidence model . . . 11
10.2 The web address finder . . . 12
10.3 Adoption rules . . . 12

11 Professor recheck **13**

12 Sending **13**
12.1 Selection . . . 13
12.2 Recipient-local send windows . . . 13
12.3 Throttling and failure handling . . . 13
12.4 Never mailing twice . . . 14
12.5 Messages . . . 14

13 Replies and follow-ups **14**
13.1 Reply tracking . . . 14
13.2 Drafting replies . . . 14
13.3 Funding follow-ups . . . 14
13.4 Meeting slots . . . 15
13.5 Resending from a second account . . . 15

14 Letters to leading researchers **15**

15 Security and privacy **15**

16 Interfaces **16**

17 Testing **16**

18 Limitations **16**

19 Outlook **17**

1 Motivation

Cold-emailing professors for a summer internship is a numbers game that is usually played badly. A student copies a few dozen names out of faculty pages, writes one template, swaps the name and sends it. Professors receive many such emails and recognise them at once. The few that get answered mention something specific from the professor's own recent work and connect it to something the student actually built.

Doing that properly by hand takes about half an hour per professor: find the person, check they still work there, read two recent abstracts, find the address they currently publish, write a short honest email, and send it at a sensible local time. Agnostos automates the mechanical parts of that work and keeps the student in the loop for the parts that matter. Every draft is written to a spreadsheet the student can read and edit before anything is sent.

1.1 Design principles

These rules hold throughout the code base:

Free first. Everything that can run locally or against a free public index does so: resume parsing, ranking, OpenAlex enrichment and discovery, email verification, the humaniser's first two stages, scheduling and reply tracking. The language model is only called for rows that survive the local filter, and the default provider (Groq) has a free tier that covers a normal run.

Honest framing. The drafting prompt tells the model the student is a second-year undergraduate whose projects are coursework-scale. Claims of results, metrics, novelty or superiority are softened or flagged (Section 9).

Never mail twice. The `Send Status` column is the record of who has been contacted. A live send refuses to start while the workbook is locked, and statuses that cannot be written are parked in a sidecar file that is treated as authoritative.

Never guess an address. There is no table mapping universities to mail domains. An address is trusted only when the name matches and the page that published it belongs to the same domain (Section 10).

Resumable. Discovery, verification, sending and reply sending can all be interrupted and re-run with the same command.

Nothing personal in the code. Paths, accounts and keys come from `.env`, which is git-ignored. Placeholder values are detected before any network call.

2 System architecture

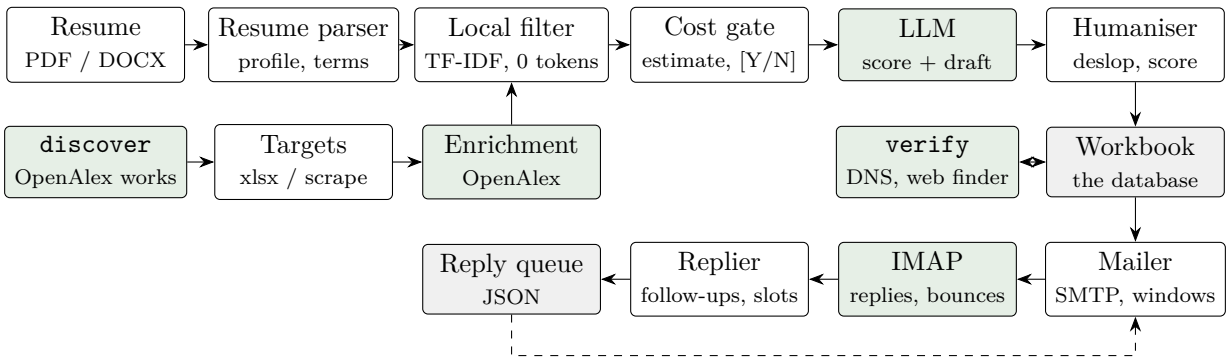


Figure 1: Data flow. Shaded boxes hold persistent state; green boxes talk to external services.

Agnostos is a single Python package, `outreach`, driven by `outreach/cli.py` (`python -m outreach.cli <command>`, or the `agnostos` console script). It needs Python 3.10 or newer. The core dependencies are `openpyxl`, `pdfplumber`, `python-docx`, `python-dotenv`, `openai`, `PyYAML`, `truststore`, `certifi` and `httpx`. Playwright is optional (Mode A scraping) and so is `ddgs` (metasearch for the address finder).

2.1 Components

Module	Responsibility
`config.py`	Settings from `.env` with defaults, provider table, model prices, placeholder detection and `ConfigError`.
`resume_parser.py`	PDF/DOCX/TXT to a `ResumeProfile`: sections, education, interests, projects, achievements, domain-term keywords.
`excel_io.py`	Mode B ingestion with header detection and about 30 header aliases; in-place result columns with backup; sending view; sidecar for locked workbooks.
`discover.py`, `batches.py`	Topic-driven discovery over the OpenAlex works index, in resumable batches.
`scraper.py`	Mode A: faculty-directory scraping with Playwright, falling back to plain HTTP.
`enrich.py`	OpenAlex author resolution and recent-publication selection, with a cache.
`focus.py`	Keyword gate that keeps outreach on the student's field.
`matcher.py`	The local TF-IDF matcher and candidate filter.
`llm.py`	Cost estimate, `[Y/N]` gate, one call per professor, capability negotiation, body repair, humaniser integration.
`humanize.py`	Deterministic deslop, AI-tell score, hard-claim and self-funding detection, rewrite prompt.
`verify.py`	Address checks (syntax, DNS, name match) and the web address finder.
`recheck.py`	Is the person still publishing, and still at this institution?
`mailer.py`, `schedule.py`	SMTP dispatch, throttling, local send windows, `-at` start times.
`replies.py`, `replier.py`	IMAP reply classification; whole-thread reading and follow-up drafting.
`gcal.py`	Meeting-slot proposal and tentative calendar holds.
`legacy.py`, `replace.py`	Letters to eminent researchers; replacing bounced contacts with a colleague.
`mailbox.py`, `convoview.py`, `stats.py`, `dashboard.py`	Sent-folder archiving, the conversation page, funnel statistics and a console dashboard.
`tls.py`	One TLS context for HTTPS, SMTP and IMAP.

2.2 The pipeline in six steps

`run` prints six numbered steps: (1) parse the resume; (2) acquire targets from the workbook (Mode B) or by scraping (Mode A), dropping rows already sent and, when `FOCUS` is on, rows outside the student's field; (3) enrich from OpenAlex; (4) apply the local filter; (5) estimate cost, ask for confirmation, then call the model once per selected professor; (6) write results back. `discover` and `verify` run before `run` when the student starts from a topic rather than a list; `send`, `check-replies` and `replies` run after it.

3 Resume parsing

`parse_resume` reads PDF (pdfplumber, `x_tolerance=1`), DOCX or plain text.

Glued text repair. Some PDF extractors drop spaces (`DeepLearning`). Let L be the number of letters, S the number of spaces and $\bar{w}$ the mean word length. The text is treated as glued when

$$\frac{S}{L} < 0.10 \quad \text{or} \quad \bar{w} > 11,$$

and only then are spaces re-inserted at lower-to-upper case boundaries, around short joiner words (`in`, `of`, `at`, ...), between letters and digits, and after punctuation. The mean-word-length test is the decisive one, because normal English averages about five to seven characters per token.

Sections. A line is a section heading when its lower-cased letters fully match one of the patterns for education, interests, projects, skills, achievements or experience, and it is either upper-case or shorter than 45 characters. Other short upper-case lines of at most six words open an unnamed section, so their content is not attached to the previous one.

Projects. Lines that start with a top-level bullet glyph open a project; following lines become its bullets; `Advised by ...` is captured as the advisor. Resumes without bullet glyphs fall back to a title heuristic: a line of at most 80 characters and 12 words that starts with a capital, has no terminal punctuation and has at least 60% of its longer words capitalised.

Keywords. The parser carries a fixed vocabulary of about 120 domain terms (`DOMAIN_TERMS`: methods, physics subfields, tools). The keyword list is every domain term present in the text, followed by other non-stop-word tokens longer than three characters that occur at least twice, capped at 45. `skills` is the list of domain terms present (up to 30). `summary_block()` turns the profile into a block of at most 2 400 characters, with up to four projects of at most 420 characters each, for the language model.

4 Target acquisition

4.1 Mode B: an existing workbook

Every sheet is scanned. The header row is the row among the first eight with the most cells that match a known header alias (`Professor Name`, `Email`, `Affiliation`, `Research Interests`, ...). Sheets without a name column are skipped, so a "Funding Programmes" tab is ignored. Columns that match no alias are folded into the notes field as `header: value`, which the model still sees. Rows are de-duplicated by email, or by name and institution when there is no email.

4.2 Mode A: directory scraping

`scraper.py` drives Chromium through Playwright. It removes the common automation tells (`navigator.webdriver`, plugins, WebGL vendor, `chrome.runtime`), randomises viewport, locale, timezone and device scale per browser context, and moves the mouse along smoothstep paths, $e(t) = t^2(3 - 2t)$, with random dwell times and scrolls. A page counts as blocked if one of the hard challenge markers appears in its first 6 000 characters, or if a softer marker appears and the visible text is at most 2 500 characters long. A blocked page is waited out once and then retried with a fresh context (and a fresh proxy exit, if a proxy is configured). `robots.txt` is honoured unless `RESPECT_ROBOTS=false`; CAPTCHAs are not solved. Delays between pages are uniform in $[4, 11]$ s by default. Names are extracted around academic titles and around email addresses (including `[at]`/`(dot)` obfuscation), generic office mailboxes are dropped, and when a directory hides addresses the scraper follows each person's profile link and matches the address by surname. Without Playwright it falls back to plain HTTP.

4.3 Topic focus

With `FOCUS=true` (the default), `run` and `send` only process rows whose research area, institution, department, recent works and topics match `FOCUS_KEYWORDS` (physics and high-energy physics stems by default). Let h be the number of *distinct* keyword matches. A single hit next to an off-field marker (clinical, cancer, protein, marketing, ...) does not count:

$$\text{in_focus} = (h \geq 2) \ \vee \ (h = 1 \wedge \neg\, \text{off_field}).$$

5 Discovery from a research topic

`discover` builds a list of professors without the student naming any university. It searches the OpenAlex *works* index for each topic and credits the people in the positions that matter.

5.1 Which papers count

Each page requests 100 works with the filter `from_publication_date:<since>-01-01`, `type:article`, `has_doi:true`, the selected countries, and institution types `education|facility`. Countries are resolved from named groups (`europe`, `dach`, `nordics`, `usa`, ...) or ISO codes, with `uk` mapped to `gb`.

Mega-collaborations. A work with more than 15 authorships is discarded. A 2 000-author detector paper names everyone and says nothing about who works on what.

Relevance floor. The first page of each topic fixes an anchor: the highest OpenAlex `relevance_score` seen, $r_{\max}$. A work is kept only if

$$r \geq \rho\, r_{\max}, \qquad \rho = 0.1.$$

Results come best-first, so once an entire page is below the floor the topic is marked exhausted.

Field anchoring. Free-text search matches words, not meaning: "geometric deep learning" also returns dentistry papers about tooth geometry. On the first page the 30 most relevant works are grouped by their primary-topic *field*, and a field is a home field when it holds at least 15% of them. Later works outside the home fields are ignored. Nothing about subjects is hard-coded; each topic learns its own fields.

5.2 Crediting authors

For each kept work, only the *first* and *last* authors are credited, and only if they have an institution with a ROR id in one of the selected countries. Each candidate accumulates: n, the number of matching papers; ℓ, how many of those had them as last author; y, the newest publication year; their titles (up to six) and topics. After every page, newcomers are hydrated in batches of 50 from the OpenAlex author records to obtain the h-index h and the works count W. A candidate is *senior* when

$$h \geq 8 \quad \text{and} \quad W \geq 20$$

(`-min-h-index`, `-min-works`). This gate removes first-year PhD students who happened to be first author.

5.3 Ranking

Senior candidates are ranked by

$$s = n + 0.6\,\ell + 2\,\max\Big(0,\, 1 - \frac{Y - y}{5}\Big) + \frac{\min(h, 60)}{60},$$

where Y is the current year. Repeat appearances dominate (someone in four matching papers is working on the topic), last-author position adds weight, recency decays linearly to zero over five years, and the h-index contributes at most one point.

Top universities. With `-top-universities N`, the region's universities are ranked by the number of papers since `-since` with more than 50 citations, grouped by institution and filtered to type `education` in the selected countries. This is a field-agnostic proxy for reputation. Discovery then only queries papers from those institutions (via `authorships.institutions.lineage`, so a lab inside a university counts), and the sort key gains a prestige bonus. For a candidate at the university of rank k among R,

$$s' = s + 5\,w\Big(1 - \frac{k-1}{R}\Big),$$

with w the prestige weight (`-prestige-weight`, default 1). The top-ranked university adds $5w$, sliding towards zero for the last one. Universities named with `-also-universities` are resolved by OpenAlex institution search and inserted at position $\min(R, 10)$ of the ranking.

5.4 Batches

The total count C is split over K batches by integer division,

$$C = qK + m, \qquad \text{quota}_i = q + [\,i \leq m\,],$$

and topics are assigned round-robin. A reused topic continues from the OpenAlex cursor the previous batch reached, so it finds new people rather than the same ones. A batch pages until it has its quota of senior researchers, the topic is exhausted, or it reaches the page cap (40 by default). A shortfall is carried into the next batch's quota.

No one is added twice: every batch skips OpenAlex ids taken by earlier batches and by the output workbook, and names are compared through an order- and accent-insensitive key (NFKD fold to ASCII, lower-case, words of two or more letters, sorted), so *Plehn, Tilman* equals *Tilman PLEHN*. Progress is stored in `<output>.batches.json` after each batch, keyed by a fingerprint: the first 12 hex digits of the SHA-1 of the JSON-encoded topics, count, batches, sorted countries, since-year, h-index and works thresholds, plus any non-default options that change which papers qualify. The page cap and `-exclude` lists are deliberately not hashed, so raising the cap continues a run instead of restarting it. Each row receives the homepage from the ROR record of the person's institution, which later seeds the address search.

6 OpenAlex enrichment

For each professor, `enrich.py` searches OpenAlex authors by name (titles stripped, five results). The first result whose affiliations or last known institutions contain a word of four or more letters from the stated institution (ignoring *university*, *institute* and *college*) wins; otherwise, in non-strict mode, the top hit is used. Only confirmed matches are cached, so a rate-limited miss is retried next time and a namesake is never cached.

The professor's works are then fetched newest first, $\max(6k, 25)$ of them for $k = 4$ publications per author (`OPENALEX_WORKS`). Works with more than 15 authors are dropped. The rest are sorted by the key

$$(-[\text{first or last author}],\ -\text{year}),$$

de-duplicated on the first 80 characters of the normalised title, and the first k are kept as `title (year) - abstract`, where the abstract is rebuilt from OpenAlex's inverted index and cut to 180 characters. Topic names (up to eight) become the professor's concepts. A 429 response sets a flag that stops further calls for the run; the pipeline continues with the text it already has.

7 The local matcher

The matcher decides, without spending a token, which professors are worth a model call.

7.1 Text normalisation and features

Text is lower-cased and passed through a synonym layer of regular expressions (for example `hep`, `lhc`, `atlas`, `cms`, `collider` → *high-energy physics*; `gnn`, `geometric deep learning` → *graph neural network*; `pinn`, `physics-informed` → *scientific machine learning*). Tokens are runs of at least three characters that start with a letter and continue with letters, digits, `+`, `#` or `-`; stop words are removed. The feature multiset of a text is its unigrams plus its adjacent bigrams. The professor side uses only text specific to that person (research area, notes, fit note, recent works, OpenAlex concepts) and deliberately excludes the department, which in Mode A is a constant inherited from the source page.

7.2 TF-IDF cosine

Let the corpus be the N_p professors plus the applicant profile, $N = N_p + 1$ documents. For a term t with document frequency $\mathrm{df}(t)$,

$$\mathrm{idf}(t) = \ln \frac{N+1}{\mathrm{df}(t)+1} + 1,$$

and terms outside the corpus get idf = 2. For a document with term counts c_t,

$$w_t = \frac{c_t}{\sum_u c_u}\,\mathrm{idf}(t)\,\beta_t, \qquad \beta_t = \begin{cases} 2.5 & t \in \text{DOMAIN_TERMS} \\ 1 & \text{otherwise,} \end{cases}$$

and the vector is L_2-normalised. The similarity is the cosine $\cos = \sum_t \hat{a}_t \hat{b}_t$ between the profile vector $\hat{a}$ and the professor vector $\hat{b}$. Computing IDF over the actual list matters: in a list of physicists the word *physics* is in every row and is correctly discounted.

7.3 Saturating domain-term bonus

Professor rows are often one-line topic strings, so coverage of the whole resume vocabulary would be the wrong measure. Instead, let M be the set of domain terms present in the profile that also

appear in the professor's normalised text, and

$$H = \sum_{t \in M} \begin{cases} 1.6 & t \text{ is a multi-word phrase} \\ 1.0 & \text{otherwise} \end{cases}, \qquad \tau = 1 - e^{-0.5H}.$$

τ rewards absolute hits and saturates: one single-word hit gives 0.39, two give 0.63, one phrase plus one word gives 0.73.

7.4 Combined score and selection

$$\text{score} = 0.45 \, \min(2.5 \cos, \, 1) + 0.55 \, \tau,$$

rounded to four decimals, and 0 when the professor text is shorter than 12 characters. Professors are sorted by score and then filtered in order:

1. no email address → `skipped: no email`;
2. score below `MIN_LOCAL_SCORE` (default 0.25) → skipped;
3. no research signal, meaning no recent works and fewer than 30 characters of personal text → skipped, because a "personalised" email written from a directory stub is generic and wastes tokens;
4. over the `MAX_PROFESSORS` cap (default 50) → skipped;
5. otherwise selected.

`-skip filter` bypasses steps 2 and 3 but still requires an email and applies the cap. The matched domain terms are passed to the model as "local keyword overlap".

8 Language-model evaluation and drafting

8.1 One call per professor

Each selected professor gets exactly one Chat Completions call that returns *both* the acceptance estimate and the email, as a JSON object with the fields `probability`, `justification`, `subject` and `body`. The system prompt contains the applicant profile, availability and funding stance, a calibration rubric (80–100 near-perfect overlap with a funded route, 60–79 strong, 40–59 adjacent or unclear funding, 20–39 weak, 1–19 wrong field), honesty rules, a list of banned phrases and the email rules (130–180 words, salutation, three or four short paragraphs, one specific thing from the professor's own recent work, one thing the student built, one clear ask, plain text). Because it is identical for every professor, providers with automatic prompt caching bill it at the cached rate after the first call. The user message is the professor's context block, capped at 900 characters (name, institution, department, research area, up to four recent works, up to eight topics, funding and route fields, matched terms).

The returned probability is clamped to $[1, 100]$. The body is repaired before anything else: a salutation *Dear Prof./Dr. Surname* is added if missing (the title is *Prof.* only if the row's name starts with Prof/Professor), a short trailing sign-off is separated, and a single block of four or more sentences is split into chunks of $\max(2, \lceil n/3 \rceil)$ sentences.

8.2 Token and cost estimate

Before any paid call, the tool prints an estimate and asks `[Y/N]`. Tokens are approximated as

$$T(x) = \max(1, \, \lfloor |x|/4 \rfloor + \#\text{newlines}(x)).$$

With n professors, system prompt $S = T(\text{system})$ and user messages u_i: if the provider caches prompts and $S \geq 1024$ (the minimum cacheable prefix),

$$\text{cached} = S\,(n-1), \qquad \text{fresh} = S + \textstyle\sum_i u_i;$$

otherwise cached $= 0$ and fresh $= nS + \sum_i u_i$. Output is estimated as $n\lfloor 0.55\, B \rfloor$, where B is the output budget (`MAX_OUTPUT_TOKENS`, default 1 100, raised to at least 4 000 for reasoning models whose names start with `o` or `gpt-5`). If the rewrite pass is enabled, $r = \max(1, \lfloor n/3 \rfloor)$ rewrites are assumed, adding $r\,(T(\text{rewrite prompt}) + 260)$ input and $230\,r$ output tokens. With prices $(p_{\text{in}}, p_{\text{cached}}, p_{\text{out}})$ in USD per million tokens,

$$\text{cost} = 10^{-6}(p_{\text{in}}\,\text{fresh} + p_{\text{cached}}\,\text{cached} + p_{\text{out}}\,\text{out}).$$

Free-tier providers (Groq) are priced at zero. Unknown models fall back to the `gpt-4.1` price $(2.00, 0.50, 8.00)$. During the run the actual cost is computed the same way from the reported usage (cached tokens subtracted from prompt tokens), and the run stops issuing calls once it reaches `BUDGET_USD` (default \$5).

8.3 Capability negotiation

Gateways differ in what they accept. The request starts optimistic: strict JSON-schema structured output, `max_completion_tokens` and an explicit temperature (0.4 for drafting; omitted for reasoning models). On a `BadRequestError` the client switches off the next capability in the fixed order

`json_schema` → `json_object` → `max_completion_tokens` (use `max_tokens`) → `temperature`,

and retries. Downgrades are remembered for the rest of the run and do not use up retries. A response cut off with `finish_reason = length` doubles the output budget, up to 16 000, and retries. Other transient errors are retried up to three times with back-off 2^a seconds after attempt a. Authentication, permission and not-found errors stop at once. Three consecutive failed professors abort the run, so a broken key does not burn the whole list. If structured output is unavailable, the parser strips code fences and falls back to the first brace-delimited span.

9 The humanising layer

9.1 Stage 1: deterministic deslop

`deslop` applies three ordered substitution tables, case-insensitively: whole phrases ("I hope this email finds you well" is removed, "I am writing to" becomes "I wanted to", "aligns with" becomes "lines up with"), overclaiming verbs swapped verb-for-verb (*achieved* → *got*, *demonstrated* → *showed*, *outperformed* → *was compared against*), and single words with the same part of speech (*utilise* → *use*, *furthermore* → *also*, *novel* removed). It then drops the middle item of the first "X, Y, and Z" triad, turns em dashes into commas, removes markdown and bullets, and fixes spacing and capitalisation. It is conservative on purpose: it only makes substitutions that cannot break grammar, because a mangled sentence is worse than a formal one.

9.2 Stage 2: the AI-tell score

For a draft x, each tell pattern i with weight w_i and $k_i(x)$ matches contributes $w_i k_i(x)$:

Tell	Weight
Unverifiable result claim (hard claim, below)	6
Boilerplate greeting ("I hope this email finds you well")	5
LLM vocabulary (delve, leverage, robust, seamless, ...)	4
Form-letter opener ("I am writing to", "I am reaching out")	4
Overclaiming (achieved, demonstrated, novel, superior, ...)	4
Em dash; markdown or bullets; performed enthusiasm	3
Alignment cliché; three-item list	3
Formal connective (moreover, furthermore, thus, hence, therefore)	2

If the draft has at least four sentences with word counts ℓ_j, mean μ and population variance σ^2, two rhythm terms are added: $+8$ if $\sigma^2 < 12$ (uniform sentence length is a strong tell of machine text) and $+5$ if $\mu > 26$. The score is

$$A(x) = \min\Big(100,\ \sum_i w_i k_i(x) + 8\,[\sigma^2 < 12] + 5\,[\mu > 26]\Big).$$

Hard claims are matched separately: a percentage, "improved/enhanced/boosted (the) accuracy/performance/results", "beat(s) the baseline/state", "best-in-class", or "resulted in improved/enhanced/superior/better". These cannot be softened safely.

9.3 Stage 3: conditional rewrite

A draft goes back to the model only when

$$A(\text{deslop}(x)) \geq \theta \quad \text{or} \quad \text{it contains a hard claim,}$$

with $\theta =$ `HUMANIZE_THRESHOLD` $= 10$ and `HUMANIZE_REWRITE=true`. The rewrite uses temperature 0.8 (higher temperature reads less uniform) and a prompt that keeps every fact and the salutation and sign-off, shortens to 120–180 words, varies sentence length and removes claims of results. The rewritten body is repaired and deslopped again and is kept *only if* its score is no higher than the original's. A failed rewrite is never fatal. Any remaining hard claim puts `UNVERIFIED CLAIM - review before sending` at the front of the `Review Flags` column, and the final score is written to `AI-Tell Score`.

9.4 Self-funding guard

Unless `CAN_SELF_FUND` is set to yes, a sentence that offers to pay one's own travel, visa, accommodation or living costs (a modal verb, a paying verb and a cost noun within a short window, or "self-fund") and contains no negation is removed. Where such a sentence begins with an availability clause ("I'm available for 8–12 weeks and can cover travel"), the availability part is kept. The same check runs again in the mailer immediately before sending, and a message that still contains an offer is skipped rather than sent.

10 Email verification and repair

10.1 Local confidence model

`check_email` starts from 100 points. An address that fails the shape check is *invalid*. If DNS shows the domain has neither MX nor A records, it is *invalid*; if the lookup itself fails, 10 points are subtracted (a failed lookup is never treated as a bad address). The local part is then compared with the person's name by $m =$ name_match $\in [0, 1]$:

m	Meaning	Penalty
1.0	a usual pattern (`first.last`, `flast`, `last`, ...)	0
0.9	truncated surname, or initials plus a numeric id	-10
0.75	surname appears intact in another form	-10
0.6	only the given name appears	-25
0.5	short initialism	-25
0.3	letters plus digits that do not match the initials	-35
0	no relationship, or a generic mailbox (`info@`, `office@`)	-55

The variant set behind $m = 1$ tries every trailing token as the surname, compound surnames joined in several ways, and particles (*van*, *von*, *de*) removed; all names are accent-folded. If the person's own page was found on a different site, 20 points are subtracted; otherwise a free-mail domain costs 30. An optional SMTP `RCPT` probe adds 10 on acceptance and marks the address invalid on a definite 550/551/553, but it is off by default because catch-all servers accept everything. The confidence is the result clamped to $[0, 100]$, and the status is *ok* when it is at least 70, *suspect* otherwise.

10.2 The web address finder

With `-repair`, the finder reads pages about the person: their profile URL (only if it has a real path, not a homepage), then the results of up to four search phrasings, beginning with `"<name> email site:<host>"` when the institution host is known. Search goes through the `ddgs` metasearch package, trying backends in turn (the last one that worked first), with a hard deadline of $3t + 10$ seconds per call ($t = 15$ s) because the library can hang; DuckDuckGo's HTML endpoint is the last resort. Requests to any single host are spaced at least 1.5 s apart across all worker threads. From each page the finder extracts plain, `mailto:`, `[at]`/`(dot)`-obfuscated, TYPO3 Caesar-shifted and Cloudflare XOR-encoded addresses, and reads the first two pages of PDFs (papers print their authors' addresses).

Each address e found on host h is scored. It is discarded if $m(e) < 0.75$, if its local part names a different given name than the person's, or if $m < 1$ and the address is on another country's top-level domain. Otherwise

$$s(e) = m + b(e) + 0.5\,[\text{academic}] + 0.25\,[\text{profile page}],$$

where the domain bonus $b(e)$ is

$$b(e) = \begin{cases} 1.0 & \text{if the address domain is the same site as } h, \\ 0.75 & \text{else if it is the same site as the institution's ROR homepage,} \\ 0.5 & \text{else if } m = 1 \text{ and the domain spells the institution,} \\ 0 & \text{otherwise.} \end{cases}$$

"Same site" means equal or one a sub-domain of the other, and "academic" means not a known free-mail host. If $m = 1$, the domain is academic and the same address was seen on two different hosts, $s \leftarrow \max(s, 2)$: two unrelated sources agree. The best address wins, and the search stops early once $s \geq 2$.

10.3 Adoption rules

A found address is only used when $s \geq 1.75$.

- Row has no address (rows from `discover`): the found one becomes the suggestion with status *suspect* and confidence $\min(90, \lfloor 55 + 15s \rfloor)$.
- Row has a different address on another site: confidence is capped at 45 and the status becomes *suspect*.
- Row has the same address: status *ok*, confidence at least 95.

With `-apply`, the suggestion replaces the row's address only if the row had none or its current address is not *ok*; the row is then re-checked. A passing address is never replaced, and the alternative stays in `Suggested Email` for a human to judge. Results with $s \geq 1.75$ are cached; misses are not, because a miss is usually a blocked search engine. Rows already sent are skipped unless `-include-sent` is given, and the mailer refuses rows marked *invalid* unless `-allow-unverified` is passed.

11 Professor recheck

`recheck` asks the question before address verification: is this person still publishing, and still there? It reads the OpenAlex author record (by id when the row has one, otherwise by a strict name search that only accepts a record whose affiliations match the row). With Y the current year and y^* the latest year with at least one work:

- *inactive* if $Y - y^* \geq 3$;
- *moved* if none of the last known institutions, nor any affiliation active in the last two years, matches the row's institution under any OpenAlex name, acronym or alternative name;
- *active* otherwise; *not found* when no record matches.

Institution names match loosely: after accent folding and removing generic words (university, institute, of, ...), two names match if they share a word, if one word contains the other (for words of four or more letters), or if a short word in one is an acronym of the other (*KIT* for Karlsruhe Institute of Technology). A row listing two places ("Nikhef / UvA") matches if either does. A failed lookup leaves the previous status untouched, and blank Institution and OpenAlex ID cells are filled in from the record.

12 Sending

12.1 Selection

The mailer considers rows with a draft and an address whose `Send Status` is not *sent* (or *review-sent* in review mode), and whose probability is at least `-min-probability`. With `-top N` the rows are sorted by probability and the best N are kept; `-limit` (default `DAILY_SEND_CAP` $= 25$) then caps real sends in this run. With `FOCUS` on, off-field rows are excluded as in `run`.

12.2 Recipient-local send windows

A window `H1-H2` (minutes $[a, b)$, with $b > a$) is evaluated in the recipient's time zone, derived from the Country column (ISO code or name). Countries spanning several zones use an override (the US maps to New York, Australia to Sydney, Germany to Berlin, and so on); others use the first `zone.tab` entry. A row may be sent at local time t when

$$a \leq 60\, t_h + t_m < b \quad \text{and} \quad (\text{weekday} \vee \texttt{-weekends}).$$

Otherwise the scheduler scans the next eight days for the first opening after t and reports the wait. Rows with no usable country are never held back. With `-wait`, the mailer closes the SMTP connection and sleeps until the earliest opening across the queue; without it, held rows are left for a later run. `-at` delays the start to a given local time (`HH:MM`, rolling to tomorrow if already past, or a full date).

12.3 Throttling and failure handling

After each successful send the mailer waits a uniformly random time in $[\texttt{SEND_DELAY_MIN}, \texttt{SEND_DELAY_MAX}] = [90, 240]$ s, using the wall clock so a sleeping laptop does not stretch the wait. If the wait exceeds 30 s, the connection is closed and reopened for the next message, because university relays drop idle sessions. Connecting is tried three times with $15a$ s pauses. A message that fails on a dropped connection is retried once on a fresh one. After a failed message the mailer pauses $\min(30, 5k)$ s for the k-th consecutive failure, and three consecutive failures stop the run so the remaining rows stay sendable.

12.4 Never mailing twice

`send -live` refuses to start if the workbook is open in Excel. Each result (status, time, Message-ID) is written back to the workbook; if the file becomes locked during a run, the updates are parked in `<workbook>.xlsx.pending-status.json` and merged on the next successful write, with later updates for the same cell winning. Rows in the sidecar count as sent for `send`, `verify` and `recheck`. Every attempt is also appended to `output/send_log.csv`. Dry runs never block a later real send.

12.5 Messages

Each message carries a fresh Message-ID on the sender's domain, an automatically appended signature built from the resume, and the files listed in `ATTACHMENTS`. A copy is appended to the IMAP Sent folder (`SAVE_TO_SENT`) so mail clients show complete threads. `-review` mails each draft to the student instead, with the real recipient in the subject; `-export` writes `.eml` files.

13 Replies and follow-ups

13.1 Reply tracking

`check-replies` logs into IMAP over TLS, selects each folder read-only and fetches with `BODY.PEEK`, so nothing is marked as read. For each sent row it searches, since one day before the earliest send, for mail from the professor's address and for mail whose `In-Reply-To` or `References` header contains the stored Message-ID. Mail from `mailer-daemon`/`postmaster` is matched to a row by the failed address quoted in the notice. Each message is classified:

- *bounced*: sender or subject looks like a delivery-status notification;
- *auto-reply*: an `Auto-Submitted` header other than `no`, `X-Autoreply`/`X-Autorespond`, `Precedence` of auto_reply/bulk/junk, or an out-of-office subject in one of several languages;
- *replied*: anything else.

A row keeps the strongest result under the order bounced < auto-reply < replied, so a later bounce never hides a real answer. A one-line snippet, stopping at the first quoted line, goes into `Follow-up Notes`; notes typed by hand are kept.

13.2 Drafting replies

`replies` reads each whole thread (inbox, spam and Sent folders, plus the original email from the workbook if Sent has no copy), orders it by date and asks the model to choose an action: *reply*, *thank* (a 40–70 word gracious close) or *none*. The prompt requires every direct question to be answered, forbids invented facts (unknowns become `[FILL: ...]`), and, when a professor says funding is unavailable, requires a reply that keeps the door open instead of a thank-you. Drafts go to `output/reply_queue.json` and are only sent once approved. If a reply says something is attached, the CV is attached; if the file is missing, the send fails rather than going out without the promised attachment.

13.3 Funding follow-ups

When a professor has raised funding and then gone quiet after the student's answer, each `replies` run may draft another follow-up, at most `FUNDING_FOLLOWUPS` = 2 per thread, and only after `FOLLOWUP_GAP_DAYS` = 2 days of silence since the student's last message. As soon as the professor writes again, the thread becomes a normal reply.

13.4 Meeting slots

The student's second message in a thread, and every funding follow-up, asks for a short call (`MEET_MINUTES` = 20) and offers three slots. `propose_slots` walks forward day by day from $t_0 = \text{now} + \texttt{MEET_LEAD_DAYS}$ (2 days) to $t_0 + 14$ days. For each weekday not yet used it tries start times on a 30-minute grid inside the professor's hours `MEET_THEIR_HOURS` (9–17 local) and takes the first start s with end $e = s + d$ such that

- $s \geq t_0$ and e is within the professor's hours;
- $[s, e]$ lies inside the student's hours `MEET_MY_HOURS` (13–23 in `MEET_MY_TZ`, Asia/Kolkata by default) on a single calendar day;
- $[s, e]$ overlaps no busy interval, where busy intervals are the slots already offered to other professors in the queue plus, if Google Calendar is connected, the calendar's free/busy blocks.

At most one slot is taken per day, so the three slots fall on different weekdays. When the email is sent, each slot becomes a tentative *HOLD* event on the student's calendar.

13.5 Resending from a second account

`resend` re-delivers the *exact* original subject and body through a second SMTP account to rows that were sent but bounced or got no answer, skipping replied and auto-replied rows, addresses known to be invalid, rows already resent (from the send log and a `Resent At` column) and rows first mailed fewer than `RESEND_MIN_DAYS` = 4 days earlier. Bounced rows go first, and `RESEND_DAILY_CAP` = 100 limits sends per day. The original send columns are never changed.

14 Letters to leading researchers

`legacy` writes a different kind of email: a short letter from an aspiring scientist to a high-impact researcher. Let A be the current year minus two. For each topic it takes the 100 most-cited works with more than 200 citations published since $A - 10$, collects the first 12 authors of each, hydrates up to 200 of them, and walks them in descending h-index order, keeping authors with $h \geq 60$ (`-min-h`) and at least two works since A (still active), up to eight per topic. People named with `-include` are added from an author search. A candidate is kept only if its topics and recent titles contain at least two distinct physics terms and no off-field marker. Recent work is taken from papers with at most 12 authors, so a letter never credits someone with a 3 000-author paper. Addresses come from the same finder as `verify`, and are rejected when the domain is in another country, is a `.edu` for a non-US institution, or shares no word, initials or abbreviated label with the institution. The model may only name a summer route from a fixed list (CERN, DESY, DAAD WISE, Mitacs, Charpak, ...) chosen from the institution and country. Each letter must be approved in the workbook before it is sent; `-approve-all` only covers addresses the finder verified.

15 Security and privacy

- **Secrets.** All credentials live in `.env`; `.env`, `credentials.json` and `client_secret*.json` are git-ignored. Values still equal to the `.env.example` placeholders are treated as unset and reported as a single `CONFIG:` message before any network call.
- **Personal data.** Workbooks, caches, logs, the reply queue and the Google token are written to `output/`, which is git-ignored, as are per-run failure reports.
- **TLS.** HTTPS, SMTP and IMAP share one context that never disables verification. It prefers `SSL_CERT_FILE`/`REQUESTS_CA_BUNDLE`, then the operating-system store through `truststore`, then `certifi`. The one exception is the optional Playwright scraper, whose browser context ignores HTTPS errors on directory pages.

- **Mailbox access.** IMAP is used read-only for reply tracking. The Google Calendar scope is the calendar scope only, and the OAuth token is cached in `output/`.
- **Politeness.** Scraping honours `robots.txt` by default, never solves CAPTCHAs and waits 4–11 s between pages. The address finder spaces requests per host, and OpenAlex calls are spaced 0.12–0.15 s apart; enrichment stops on a 429.
- **What gets sent.** Nothing is sent without `-live`. The cost gate, hard-claim flags, the self-funding guard, the invalid-address block and the consecutive-failure abort all sit between a draft and a professor's inbox.

16 Interfaces

CLI. `profile`, `models`, `run`, `discover`, `verify`, `recheck`, `send`, `stats`, `check-replies`, `replies`, `conversations`, `archive-sent`, `resend`, `legacy`, `replace-failed`, `calendar-auth` and `tls-check`.

Workbook. Result columns appended in place (after a timestamped backup): `Local Match Score`, `Probability of Acceptance`, `Probability Justification`, `Email Subject`, `Custom Email Draft`, `AI-Tell Score`, `Review Flags`, `Email Status`, `Email Confidence`, `Email Notes`, `Suggested Email`, `Profile Status`, `LLM Status`, `Send Status`, `Sent At`, `Reply Status` and `Follow-up Notes`. Re-running refreshes these columns instead of duplicating them, and blank model results never overwrite cells edited by hand.

Files. `output/send_log.csv`, `output/reply_queue.json`, `output/conversations.html`, `<output>.batches.json` and the OpenAlex, discovery and address caches.

Providers. Any OpenAI-compatible endpoint: Groq (default, `openai/gpt-oss-120b`), OpenAI (`gpt-4.1`) or a gateway set by `OPENAI_BASE_URL`.

17 Testing

Two suites run on a fresh clone with no `.env` and no API key, using synthetic fixtures. `tests/smoke_test.py` is fully mocked: resume parsing, Excel round-trips, filter behaviour, cost arithmetic on free and paid providers, a mocked model pass, abort on repeated failures, body repair, the humaniser, idempotent column writes, attachments, resume safety for interrupted sends, `.eml` export, scraper extraction and block detection, address verification, batch planning and fingerprints, institution matching for `recheck`, and the rule that `-apply` never replaces a passing address. `tests/openai_stub_test.py` runs the real `openai` SDK against a local HTTP stub and checks request shape, cached-token accounting, temperature omission for reasoning models, truncation handling and capability downgrade. CI runs both on Python 3.10, 3.12 and 3.13.

18 Limitations

- The acceptance probability is the language model's guess against a rubric, not a statistical estimate. It is useful for ordering, not as a frequency.
- The local matcher is lexical. Synonyms and the domain vocabulary help, but two descriptions of the same work in different words can still score low. The vocabulary and the default focus keywords lean towards physics and machine learning.
- OpenAlex author disambiguation is imperfect: one person can be split across several ids and a namesake can be merged in. Discovery keeps one record per name; recheck uses strict institution matching to limit the damage.

- The address finder depends on public search engines that block scrapers unpredictably. A miss is usually a blocked search, not proof that no address exists. Repair takes 20–40 seconds per professor.
- Reply classification is rule-based and can mislabel unusual auto-replies.
- Time zones come from the country, so a professor in a country with several zones is scheduled in its main one.
- The token estimate uses four characters per token and a fixed rewrite rate; the budget cap itself uses the provider's reported usage.

19 Outlook

Natural next steps are an embedding-based matcher that runs locally beside the lexical one, a field-neutral default vocabulary derived from the resume itself, per-institution time zones from ROR coordinates, and learning the reply classifier from the student's own labelled threads. A hosted web version of the tool is developed separately as Agnostos v3.

Agnostos is MIT-licensed. It drafts; the student reads, edits and signs every email.