elife-claim-trees

A claim-extraction pipeline for eLife papers

Zach Mainen — Mainen Lab, Champalimaud Foundation *With contributions from agentic collaborators*

2026-05-11 · Documentation: `https://zmainen.github.io/elife-claim-trees/docs/`

Agenda

1. **Why** — the problem, the bet
2. **Methodology** — the 8-step claim induction process and the schema
3. **Architecture** — the pipeline, the agents, the review gate
4. **Demo** — Headley 2026 walked through, end to end
5. **Validation** — round-trip scoring on 10 papers
6. **How the system improves** — measuring prompt changes before they ship
7. **Working together** — where eLife could plug in (API, UX, algorithm, scaling) + UK grant + next steps

Q&A throughout — interrupt freely.

1. WHY

What a structured claim corpus actually buys us

Scientific publishing has a corpus problem

Each paper is a closed unit. Read once, then mostly forgotten. The knowledge it carries doesn't compose with the rest of the field.

	TODAY	WHAT IT SHOULD BE
Reading a result	"Find the figure, read the caption, hope it's clear"	Click on a claim, see the analysis script, the data, the dependencies
Cross-paper search	"Grep abstracts; trust your reading"	Query a graph: "what does the field say about X?"
Reproduction	"Re-read the paper, find the GitHub link, hope the data is online"	Each claim links to a verifiable computation
Meta-analysis	$O(N^2)$ re-reading	O(log N) graph lookup once the structure exists
When a result is wrong	The paper stands; corrections rarely propagate	Invalidity propagates through the dependency graph

The corpus *accumulates* but doesn't *compose*. The structure that would make composition possible is missing from how we publish.

The unit question — what's the right grain?

GRAIN	PROBLEM
Statistics (one p-value, one effect size)	Too granular. A statistic without context is meaningless.
Figures (one figure per claim)	Too coarse. One figure typically contains multiple sub-claims via its panels.
Sections (results section per paper)	Way too coarse. Conflates dozens of claims into one prose block.
Papers (one paper, one identity)	The unit publishing currently uses. Hides the actual claims.
Sentences (NLP claim extraction)	Too granular and too noisy. Loses causal structure.
➜ Panels	One analysis. One data source. One result. **The natural unit of reproducible science.**

> "The unit of publication is not a figure or a statistic — it is a computer program."
> — Kenneth Harris, formulating the project's thesis at the initial eLife meeting

A claim is a computation: data (in), analysis script (transform), result (out). The panel is where one computation lives.

What a claim looks like

A single declarative proposition, anchored to a panel, linked to its computation and to other claims:

```
claim: Doubling distal dendritic inhibition reduces somatic firing rate
       from approximately 5.5 Hz to approximately 0.2 Hz.
panel: fig4, fig5
analysis: scripts/Fig4.ipynb
data: https://datadryad.org/dataset/doi:10.5061/dryad.v6wwpzhb8

requires:    [l5-model-single-cell-scope]
supports:    [hypothesis-distinct-compartmental-roles]
dissociates-with: [perisomatic-inhib-drops-firing-07hz]

reproductions:
  - status: verified
    figure: <reproduced figure URL>
```

Five things stitched together: **a proposition**, **a panel**, **the computation that produced it**, **the claims it depends on**, and **whether it reproduces**.

Important — claims need not be explicit in the paper's prose. The empirical findings usually are; the *implicit hypothesis* the paper bets on, and the *predictions* deduced from its modelling structure, often live only in the architecture of the experiment. Recovering implicit claims is one of the harder pieces — we'll come back to it in the architecture section.

What becomes possible

Once panel-level claims are first-class entities in a graph:

CAPABILITY	WHAT IT ENABLES
Claim-level peer review	Reviewers see propositions, not paragraphs. Reviewer disagreements localize to specific claims.
Cross-paper search	"What does the field say about X?" becomes a graph query, not a literature review.
Reproducibility infrastructure	Every claim links to its analysis. Reproduction status is part of the record.
Invalidity propagation	When claim B fails to reproduce, traverse the graph to find every claim that `requires` B across the corpus.
Living reviews	A review paper that updates as new claims enter the corpus.
Anti-hallucination	Citations resolve to specific claims with verified DOIs, not just paper-level pointers.
Tractable meta-analysis	The unit of comparison is structured; aggregation is no longer a re-reading exercise.

The Library Theorem (`arXiv:2603.21272`) formalizes the gain: indexed retrieval is *exponentially* cheaper than unindexed scanning. Structured claims are the index.

The fine-grained citation graph

The current paper-citation graph is what we have. The claim-level dependency graph is what we want.

	TODAY (PAPER-LEVEL)	TOMORROW (CLAIM-LEVEL)
Unit of citation	"Paper A cites Paper B"	"Paper A's claim 5 `requires` Paper B's claim 12"
What a citation tells you	"B is somehow relevant"	"Specifically these propositions stand in this typed relationship"
Tracing a result	Read both papers and infer which result	Direct edge to the cited claim
Invalidity propagation	Manual; corrections rarely propagate	Graph traversal: when claim B-12 fails, find every claim that requires it
Field-level synthesis	Re-read the literature	Query the graph
Graph density at corpus scale	~10 citations per paper	~50-100 typed edges per paper × N papers; an actual *structure* of what's known

This is the long horizon. Today's pipeline produces the per-paper claim files; the cross-paper edge resolution is a separate downstream system that builds on `verify-refs` outputs.

But the value compounds: each paper added to a claim-graphed corpus enriches the cross-paper structure. At N=10 it's a curiosity. At N=1000 it's a research instrument. At N=100k it's the structure of a field.

Claim-level peer review

Even before corpus-scale graph effects, claim-level structure transforms peer review *today*. QED Science (Yousef Hashash et al.) has been pioneering this direction; the empirical case:

	PAPER-LEVEL REVIEW (TODAY)	CLAIM-LEVEL REVIEW (WITH STRUCTURE)
Where reviewer comments attach	The whole paper	A specific claim
When reviewers disagree	Hard to tell what they actually disagree about	Disagreement localizes to specific propositions
Editor adjudication	Synthesizing prose comments across 2-3 reviewers	Reading per-claim positions; the disagreement shape is visible
Tracking issues across rounds	Re-read the prior round	Each claim has a history; what was contested, how it changed
Author response	Wall of prose addressing wall of prose	Specific responses to specific contested claims

This is the **most concrete near-term win** — it doesn't require corpus-scale adoption, it doesn't require authors to change how they write. eLife can offer claim-level review as an editorial layer today; the extraction pipeline produces the structure reviewers and editors work with.

The bigger picture — claim-native publication

Three layers of work, each a step toward science where claims rather than papers are the fundamental unit of knowledge:

LAYER	WHAT	STATUS
1. Extraction	Convert the existing corpus to claim-level structure. The pipeline this talk is about.	**Built and working**
2. Authoring	Tools that help authors compose papers as claim units from the start. Reduces the conversion need over time; produces claim-native output directly.	Sketched; not built
3. Sandbox / agent-driven	Agents themselves publish at claim-level granularity — a long-horizon experiment in scientific knowledge production at the unit the methodology actually works at.	Vision; for the longer arc

The pipeline this talk is about is layer 1. It exists *because* the existing corpus needs handling — for as long as authors publish in monolithic papers, conversion is needed. Layer 2 reduces that need over time. Layer 3 is what becomes possible when claim-native is the default.

Where eLife fits — two integration modes

Two ways the pipeline integrates with eLife's editorial infrastructure. Both use the same extraction pipeline; what differs is timing and authority.

MODE	WHAT HAPPENS	WHEN TO SHIP
Author-approved (per submission)	Author opts in; system extracts claims; author reviews and approves the claim graph before publication	Longer-term; requires editorial workflow integration
Automatic public-domain (corpus build)	System processes published papers without author involvement; surfaces claim graphs as an editorial layer for reviewers, editors, and readers	**Today** — eLife can ship this on the existing public corpus

eLife is uniquely positioned for both:

- **Open peer review** already published — claim-level structure makes review legible and traceable
- **Open data deposits** mandated — every claim already has the computation it depends on
- **Editorial workflow** that supports structural innovation
- **Technical team capacity** to operate and extend infrastructure (that's why we're talking today)

The methodology is documented; the corpus exists; the pipeline works. The decision is which mode to ship first and on what timeline.

Status — a working draft, not a finished product

What exists:

- Pipeline built, all 8 methodology steps + Step 4.5 reviewer extension
- 10-paper public eLife corpus extracted via round-trip; empirical performance documented
- 32-page documentation site; 5 prompt files version-controlled; round-trip evaluation harness

What's still in iteration:

- 2 of 10 papers (rozak — methods paper, kolb — sensor engineering) score below role threshold; need targeted prompt variants
- Multi-panel handling needs another iteration cycle (panel agreement at 60%, threshold 90%)
- Step 6 (dependency mapping) is scaffolded; analyst work for now
- Methodology fallback chain (PDF → API → web fetch) only PDF path implemented
- Authoring tools (layer 2 above) not yet started

eLife participation in the iteration would be valuable. Different paper types, different domains, different editorial priorities will surface failure modes the lab corpus didn't.

This is the honest framing: the system **works** at the empirical thresholds we set; it is **not** a finished product. Adopting it means adopting the iteration discipline alongside the pipeline.

The implementation problem

OK — claims are the right unit. Structured corpora unlock these capabilities. eLife is the right partner.

Now the operational reality:

> Manual extraction of one paper's claim graph takes a curator **~4 hours**.
> The 12-paper curated reference corpus took **weeks**.

For journal scale — 100s of papers/year, eventually thousands — the bottleneck is **per-paper analyst time**.

What's needed:

- A pipeline that produces **curator-quality output**
- Without requiring **per-paper curator review**
- That **integrates with the existing methodology**, not replaces it
- That can be **operated by your team**, not just the originating lab

The rest of this talk is about that pipeline.

The naive approach fails in characteristic ways

A single LLM reading the whole paper exhibits four predictable failure modes:

FAILURE MODE	WHAT IT LOOKS LIKE
Quantitative hallucination	"approximately 5 Hz" becomes "5.2 ± 0.3 Hz" with invented precision
Wrong panel assignment	Claims attached to figure setup panels, not result panels
Missed methodological structure	Scope conditions, controls, analytical capabilities are flattened
Missed deductive structure	The hypothesis-prediction-test scaffolding curators infer from modelling decisions disappears

We tested these empirically. Each is real. Each requires its own architectural fix.

The bet

Three independent agents, each constrained to a focused slice of the paper, then a reconciliation step, then a structural-inference review pass.

Each piece exists because of a specific failure mode it addresses. None is decorative.

2. METHODOLOGY

The 8-step claim induction process and the schema

The 8-step methodology

#	STEP	WHAT HAPPENS
1	Prepare	Locate paper + code + data; map figure structure
2	Abstract scan	Identify 2-4 top-level claims
3	Three independent extractions	Results-reader, Caption-reader, Structure-reader
4	Reconciliation	Fold three lists into one confidence-tagged draft
4.5	**External reviewer ★**	**Recover deductive layers from prose (HAAK extension)**
5	Review gate	Analyst approves before write
6	Dependency mapping	Typed edges between claims (14 edge types)
7	Write claim files	UUIDs + frontmatter + body
8	Verify	Per-paper `verify.py` against deposited code/data

Documented at `docs/method.md` § 3 in the corpus repo. The system implements all 8, plus an extension at Step 4.5.

A claim, schematically

```
---
uuid: 26819b09-5b20-4c90-b9f3-8bdd29a2a57c
slug: distal-inhib-drops-firing-02hz
claim: Doubling distal dendritic inhibition reduces somatic firing rate
       from approximately 5.5 Hz to approximately 0.2 Hz.
claim-type: empirical
role: empirical
panel: fig4, fig5
epistemic: strong

tests:
  - prediction-distal-dendritic-spike-mechanism
dissociates-with:
  - perisomatic-inhib-drops-firing-07hz
requires:
  - l5-model-single-cell-scope

assertions: [...]   # paper-panel-analysis tuple
reproductions: [...]  # verification record
---
```

One file per claim. YAML frontmatter for structure; markdown body for prose.

What the curator does, what the system does

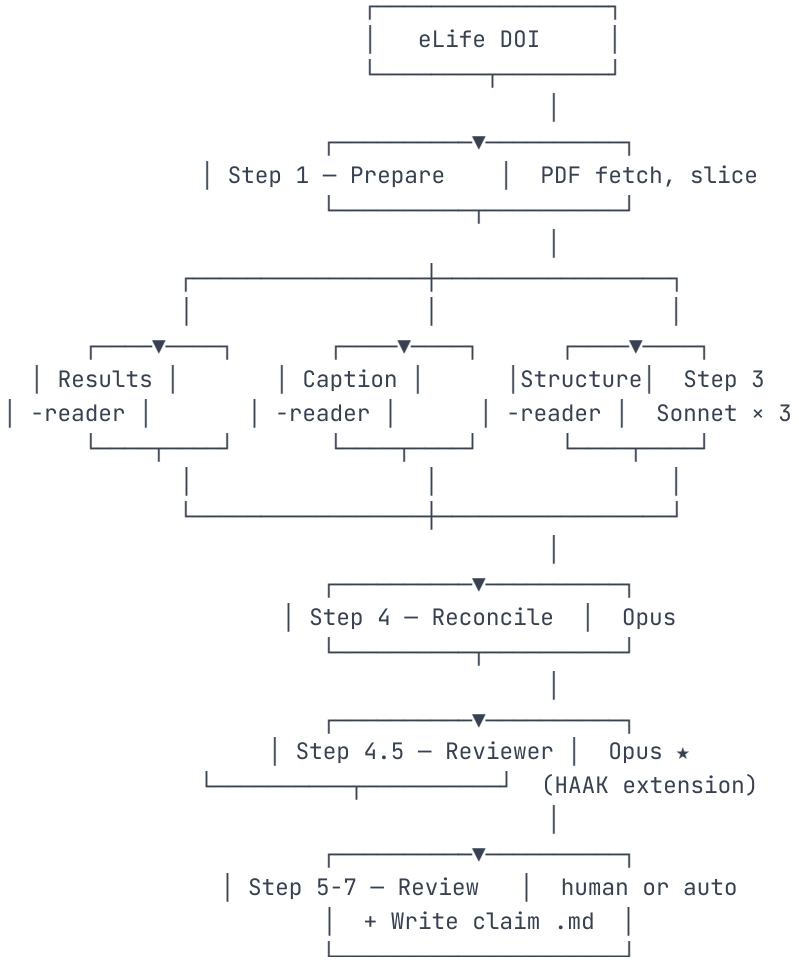
WORK	CURATOR VALUE
PDF fetching, text slicing	None — system handles deterministically
Surface extraction (claims from prose)	None — agents are reliable readers
Quantitative grounding (verbatim quoting)	None — Caption-reader prompt enforces
Cross-agent reconciliation	Low — Opus does this well
Surface roles (`empirical` , `interpretive`)	Low — system handles ~83% on average
Functional roles (`control` , `synthesis`)	**Medium** — the role's about *function*, not content
Edge mapping (Step 6)	**High** — currently scaffolded, analyst work
Epistemic assessment	High — requires reading argument structure
`displayClaim` / `shortClaim` (auxiliary)	High — needs domain feel
Concept tagging (controlled vocabulary)	High — needs your vocab

The system covers the mechanical layer; the curator's value concentrates in the **intellectual layer** at Step 5 (and where Step 5 is skipped, in post-write review).

3. ARCHITECTURE

The pipeline, the agents, the review gate

The pipeline at a glance



The three-agent partition — why three?

Each agent reads only its slice. Each catches what the others can't.

AGENT	READS	CATCHES
Results-reader	Abstract + results prose	Framing, hypotheses, predictions, synthesis claims
Caption-reader	Figure captions, panel-by-panel	Panel-level numerics, exact quantitative values, panel anchoring
Structure-reader	Methods + supplements + code	Methodological capabilities, scope conditions, control documentation

No agent sees another's output before submitting. They are independent witnesses. Their disagreement and convergence is itself the signal.

Reconciliation — three lists become one

For each claim in the three lists:

- **All three agents agree** → `confidence: high`
- **Two agree, one differs** → `confidence: contested`, flag the discrepancy
- **Only one surfaced** → `confidence: single-source`

Single-source is the largest bucket (~half) — and that's expected. The Structure-reader's methodological claims, the Caption-reader's panel detail, the Results-reader's synthesis claims — each agent surfaces things the others don't have a path to.

The reconciler is one Opus call. The hard problem it solves is **semantic matching across phrasings**. Embedding similarity is too noisy; literal string match is too strict; Opus does the alignment.

Step 4.5 — the external reviewer

The methodology says Step 5 is a hard human-review gate. Operationally — for batch, for journal-scale — that doesn't work.

The external reviewer is an Opus pass that **substitutes for the curator at the review gate**. It reads the paper plus the reconciled draft and addresses **seven systematic biases** the prose-level extraction misses:

1. Prediction role under-coverage
2. Hypothesis role under-coverage
3. Multi-panel claim collapse
4. Synthesis vs interpretation confusion
5. Hypothesis-to-prediction over-shifting (added Phase F)
6. Control role under-recognition (added Phase F)
7. Literature-context under-recognition (added Phase F)

This is the system's **most novel architectural decision**. Empirically: lifts role classification from ~65% to ~83% mean across the 10-paper corpus.

The review gate — four modes

MODE	WHAT HAPPENS	COST	TIME	WHEN
`interactive`	Open draft in `$EDITOR`; analyst edits	$5 + analyst	5 min CLI + 15-30 min analyst	Production single-paper, full quality
`external`	Opus pass; no human in loop	$7	10-15 min	**Recommended for batch**
`auto-approve`	Skip review; write as-is	$5	5 min	Tests, demos, when post-write review follows
`dry-run`	Print only; no writes	$5	5 min	Inspecting before committing

The methodology's design (`interactive`) is the curator-and-agents loop. The system's design (`external`) lets the loop close without a curator while preserving most of the quality.

4. DEMO

Headley 2026, end to end

One DOI in

```
elife-extract extract \
  --doi 10.7554/eLife.95562 \
  --corpus-dir /tmp/demo \
  --output-dir /tmp/demo/out
```

That's it. The CLI handles everything from there.

What happens next:

1. Fetch PDF from eLife CDN (cached for re-runs)
2. Slice into abstract / results / captions / methods
3. Three Sonnet calls in sequence
4. Opus reconciliation
5. Write draft JSON to `out/`

Wall time: **~5 minutes**. API cost: **~$5**.

What the system tells you

```
═══ Step 1 — Prepare ═══
  doi    = 10.7554/eLife.95562
  slug   = headley-2024-spatially-targeted-inhibitory
  slices = abstract:7184c results:88354c captions:87918c methods:26173c
  panels = 37 detected

═══ Steps 2-3 — Three-agent extraction ═══
  results-reader   →  48 candidate claim(s)
  caption-reader   →  47 candidate claim(s)
  structure-reader →  17 candidate claim(s)

═══ Step 4 — Reconciliation (claude-opus-4-6) ═══
  draft has 76 claim(s):
    high             34
    contested         5
    single-source    37

═══ Output ═══
  draft  → out/draft-headley-2024-spatially-targeted-inhibitory.json
```

48 + 47 + 17 = 112 candidates → 76 reconciled. **No claim files written yet** — Step 5 review gates the write step.

Then the review step

```
elife-extract write \
  --draft out/draft-headley-2024-spatially-targeted-inhibitory.json \
  --corpus-dir /tmp/demo \
  --review-mode external
```

```
═══ Step 5 — Review gate (external) ═══
  external reviewer: calling claude-opus-4-6...
  external review: 76 → 86 claims after revision

═══ Steps 6-7 — Write claim files ═══
  wrote 87 files (86 claims + 1 index.md)
```

The reviewer added **1 hypothesis + 6 predictions** (the implicit deductive layer Headley structures through its modelling) plus 3 role corrections.

Wall time for write: **~5 minutes**. Cost: **~$2**.

What you have at the end

```
/tmp/demo/headley-2024-spatially-targeted-inhibitory/
├── index.md                                          # paper metadata
├── all-results-derive-single-cell-compartmental.md   # 86 claim files
├── beta-bidirectional-dendritic-control.md
├── distal-dendritic-inhibition-decreased-nmda.md
├── ...
└── (and 82 more)
```

Each claim file is a complete, schema-conformant `.md` document with frontmatter and body.

Total per-paper cost: ~$7. Total per-paper time: ~10-15 minutes.

Compare to the methodology's hand-extraction baseline of ~4 hours of curator time per paper.

5. VALIDATION

How we measure quality, and what the numbers say

The round-trip methodology

For each paper that has a curated reference:

1. Run the pipeline on the paper's DOI
2. Score the CLI output against the curated reference via an Opus matcher
3. Compute per-metric scores: recovery, role agreement, panel agreement

```
elife-extract evaluate \
  --reference-dir ~/Projects/mainenlab/elife-claim-trees/claims \
  --work-dir /tmp/eval \
  --all \
  --review-mode external
```

The 10 public eLife papers in the curated corpus are the test set. Each sweep takes ~85 minutes and ~$100. Per-paper scorecards persist for diagnostic inspection.

Aggregate metrics — current baseline (v2)

METRIC	MEAN	MEDIAN	THRESHOLD	STATUS
Claim recovery	**96.9%**	100.0%	≥ 80%	✅ PASS
Role classification	**83.1%**	84.3%	≥ 75%	✅ PASS
Panel assignment	60.9%	58.6%	≥ 90%	❌ FAIL

10 of 10 papers succeeded. Recovery and role both clear thresholds; panel is the documented multi-panel collapse problem (a Caption-reader prompt iteration target).

Median recovery is 100% — every reference claim has a corresponding CLI extraction on half the corpus.

Per-paper detail

PAPER	N_REF	N_CLI	RECOVERY	PANEL	ROLE
artiushin (atlas)	17	79	**100%**	59%	94%
bouyeure (fear RSA)	30	66	**100%**	47%	90%
ejdrup (dopamine)	25	83	96%	58%	83%
gadeke (guilt insula)	27	65	**100%**	48%	85%
headley (rhythms)	26	89	**100%**	**81%**	92%
kammer (foveal)	23	43	96%	**91%**	82%
kolb (iGABASnFR2)	20	70	**100%**	75%	70%
rozak (neurovascular DL)	23	72	87%	60%	55%
scheller (self-prioritization)	23	59	**100%**	48%	83%
wengert (KCNC1)	31	91	90%	43%	**96%**

Only two papers below role threshold: rozak (methods paper, role 55%) and kolb (sensor engineering, role 70%). Both candidates for `--prompt-variant`.

6. HOW THE SYSTEM IMPROVES

Measuring every prompt change before it ships

Measuring changes empirically — not guessing

Every change to the system goes through the `evaluate` harness before deployment. This is the discipline: prompt revisions become hypothesis tests, not guesses.

```
1. Establish baseline    evaluate --all → aggregate-baseline.md
2. Make the change       (prompt revision, model swap, ...)
3. Re-run evaluate       evaluate --all → aggregate-iter1.md
4. Diff aggregates       compare metrics; per-paper deltas
5. Decide                ship if net win; revert if regression
6. Inscribe              worklog with measured delta
```

Cost per full sweep: **~$100, ~85 min**. Per single-paper iteration: **~$10, ~15 min**.

For targeted changes, the single-paper iteration is the right pattern. For corpus-wide validation, the full sweep is the right pattern.

A worked example: kammer iteration

Problem. First sweep: kammer scored 57% role — worst paper in the corpus.

Diagnostic. Per-claim matches showed 9 role failures, clustered:

- 3 cases of CLI=`empirical` vs ref=`control` (functional role missed)
- 2 cases of CLI=`prediction` vs ref=`hypothesis` (over-shifted)
- 1 case each of various judgment-call mismatches

Hypothesis. Add three new biases to the reviewer prompt:

- **Bias 5**: Don't reclassify hypotheses as predictions
- **Bias 6**: Recognize empirical claims that function as controls
- **Bias 7**: Recognize implicit literature-context claims

Cost of test: ~$10 (single-paper re-run + matcher).

kammer iteration — result

METRIC	BEFORE	AFTER	DELTA
Recovery	91%	91%	unchanged (expected)
Panel	52%	48%	-4 (within noise)
Role	**57%**	**71%**	**+14 points**

Single iteration: 14-point gain on the targeted paper.

But the question wasn't just "does it help kammer?" It was "does it generalize?"

So we ran the full 10-paper sweep with the iterated prompt. Cost: **~$100**.

kammer iteration generalized — v1 → v2 across the corpus

METRIC	V1 BASELINE	V2 ITERATED	Δ
Recovery (mean)	95.3%	**96.9%**	+1.6
Recovery (median)	98.0%	**100.0%**	+2.0
Role (mean)	78.1%	**83.1%**	**+5.0**
Role (median)	76.4%	**84.3%**	**+7.9**
Panel (mean)	54.7%	**60.9%**	+6.2

Material per-paper deltas:

- **kammer**: role 57% → **82%** (+25), panel 52% → **91%** (+39)
- **headley**: panel 54% → **81%** (+27)
- **bouyeure**: role 80% → 90% (+10)
- **rozak**: role 65% → 55% (-10) — only material regression

The iteration is a measurable net win. v2 is now the default.

7. WORKING TOGETHER

Where you could plug in, and what we could build together

Situating in the broader movement

This work doesn't sit in isolation. It's one piece of a wider movement toward modular, structured, machine-actionable scientific publishing.

INITIATIVE	WHAT IT BRINGS	WHERE THIS WORK FITS
OXA / Curvenote (oxa.dev)	Modular publishing toolchain — MyST authoring, JATS-XML output, structured components	A natural downstream consumer for claim-level structure; OXA's modules could embed claim graphs
elifePathways (elifepathways.org)	Beyond-the-journal initiative — research routes that don't require a single monolithic paper	The unit of publication elifePathways needs is closer to claims than to papers
Claim-level peer review (QED Science et al.)	Structured review at the claim level, not the paper level	The extraction pipeline produces the structure reviewers and editors work with
Reproducibility infrastructure (Dryad, Zenodo, GitHub-deposited code)	Each claim's computation is already deposited and citable	We close the loop: claim ↔ deposited computation ↔ verifiable result

The pieces exist independently. The movement is the convergence. The four questions you raised are exactly where convergence happens operationally.

Four areas where we could collaborate

The questions Damian raised, with what already exists and where joint work would unlock more:

QUESTION	WHAT EXISTS TODAY	WHERE JOINT WORK MATTERS
Q1. Data integration / API	File-based JSON + markdown output; CLI is process-safe	Wrap as a service for editorial-system integration; standardize on modular publishing formats
Q2. User experience	Per-paper claim viewer (DAG, drawer, abstract↔claim mapping)	Editorial review surface; cross-paper browser; author tools; discovery UX
Q3. Extraction algorithm	5 prompts + 7-bias reviewer; `evaluate` harness; v2 measured at 96.9% recovery / 83.1% role	Prompt iteration at scale; model substitution; edge mapping (Step 6); reference-set growth
Q4. Scaling	$7/paper, ~15 min/paper sequential; 10-paper sweep validated	bioRxiv-scale economics; Anthropic Batch API; per-paper-type variants; rate-quota engineering

The next four slides go into each in detail.

Q1 — Data integration / API

What exists today. The pipeline produces three layers of output, all file-based:

- **Reconciled draft JSON** — `out/draft-<slug>.json`. Pydantic-validated; the wire format between Step 4 and Step 5.
- **Reviewed draft JSON** — `out/<draft>.reviewed.json`. Post-Step-4.5 revision; the input to write.
- **Claim files** — `<corpus>/<slug>/{index.md, <claim-slug>.md, ...}`. Markdown with YAML frontmatter, schema-conformant, version-controllable in git.

Any consumer can read the file system or git repo directly. The intermediate JSON is a stable wire format.

Where joint work matters.

PATH	EFFORT	USE CASE
HTTP API wrapping the CLI	~2 weeks (FastAPI + the existing package)	Editorial-system trigger per submission
Streaming pipeline (Kafka / queue + worker)	~1 month	High-throughput batch on a corpus
Modular publishing format adapter (MyST/JATS-XML emitter from claim files)	Per-format spec	Direct ingestion into OXA / similar toolchains
Schema standardization with OXA	Ongoing alignment	Claims become a portable unit across the movement

The system is designed for embedding. The next step is choosing **which embedding** matters most for eLife's editorial workflow.

Q2 — User experience around claim-tree data

What exists today. The eLife claim-trees site already renders per-paper claim graphs:

- **ClaimDAG** — interactive dependency graph (React Flow); shows the typed-edge structure between claims
- **ClaimDrawer** — claim detail panel with evidence, panel anchor, reproductions
- **ClaimStructure** — hierarchical tree view (hypothesis → predictions → tests)
- **AbstractAlignment** — abstract sentences mapped to specific claims
- **ArgumentFromGraph** — synthesis reconstructed from the graph alone (a diagnostic against the abstract)

This is one paper at a time. The cross-paper UX — search, browse, traverse — doesn't exist yet.

Where joint work matters.

UX SURFACE	WHAT IT WOULD DO	WHO BENEFITS
Editorial review surface	Reviewers see claim-level structure during review; comments attach to specific claims	Reviewers, editors
Cross-paper claim browser	Search and navigate claims across the corpus	Researchers, editors
Claim-level discovery	"What does the field say about X?" as a typed graph query	Researchers, the field
Author composition tools	Help authors structure submissions as claim-units (the "authoring" layer from earlier)	Authors
Dependency-graph visualization at scale	Cross-paper traversal — invalidity propagation, field-level synthesis	Editors, meta-analysis
Embedding into OXA / elifePathways	Claim-level structure as a first-class component in modular publishing	Whole movement

Q3 — Where data science can help with the extraction algorithm

The leverage points. The system is built for empirical iteration. Six places where a data science team contributes meaningfully:

LEVERAGE POINT	WHAT IT LOOKS LIKE	WHAT DATA SCIENCE BRINGS
Prompt iteration at scale	Add bias rules; A/B test variants via `evaluate`	Larger reference sets; more diverse paper types
Model substitution	Test cheaper models (Haiku) for some agents; measure quality drop	Cost-quality Pareto frontier exploration
Reference-set expansion	Curated 12-paper corpus is the validation baseline	Each new curated paper sharpens validation; eLife scale needed
Step 6 (edge mapping)	Currently scaffolded — analyst work	LLM-suggestion pass + validation methodology
Multi-panel handling	Documented limitation (panel agreement at 60%)	Targeted Caption-reader / reconciler iteration
Domain adaptation	`--prompt-variant` infrastructure exists	Variants for methods papers, observational studies, non-neuroscience

The discipline matters. The `evaluate` harness makes every change measurable: change → re-run → diff aggregate → ship or revert. This isn't prompt engineering as guesswork; it's prompt engineering as hypothesis testing.

For data scientists familiar with ML evaluation pipelines, this should feel native. The harness is ~300 lines of Python; the prompts are markdown files in version control. Low ceremony.

Q4 — Scaling to eLife, bioRxiv, and beyond

Today's economics.

CORPUS SIZE	TIME (SEQUENTIAL)	COST (DEFAULT CONFIG)
1 paper	~15 min	~$7
10 papers (validated)	~2.5 hrs	~$70
1,000 papers (eLife-scale)	~10 days sequential, ~3 days at 3-way parallel	~$7K
150,000 papers (bioRxiv-scale)	impractical at default config	**~$1M** at default config

Where the economics improve.

OPTIMIZATION	COST REDUCTION	ENGINEERING EFFORT
Anthropic Batch API (24-hr turnaround)	~50%	~1 week to wire in
Cheaper extraction agents (Haiku for some agents)	~40%	Already supported via `--model-*` flags; measure with `evaluate` first
Concurrent extraction (asyncio for the 3 agents)	~2x latency, no cost change	~1 day
Per-paper-type triage (skip pipeline for atlas/methods papers; lighter variant)	~20-50% on those types	Per-variant; measurable via `evaluate`
Anthropic prompt caching for repeated-context calls (the reviewer sees the paper	~30% on Opus calls	~1 day

The UK grant opportunity

The grant call (or future calls in the same area) features structured-publishing technology — exactly this kind of work.

What this project could contribute:

- **Concrete deliverable**: a working extraction pipeline with measured performance across a 10-paper validated corpus. Not vapourware; demonstrable.
- **Methodology**: documented 8-step process from `docs/method.md` § 3, plus the system's structural-inference extension at Step 4.5.
- **Validation harness**: `evaluate` produces measurable progress on each iteration. The kind of empirical discipline grant reviewers want to see.
- **Open infrastructure**: prompts in version control, schema documented, code Python-stack-standard. No proprietary lock-in.
- **Convergence with the movement**: aligned with OXA, with elifePathways, with QED Science. The proposal can frame this as ecosystem, not solo.

What we can offer the proposal. Methodology details, validation results, integration architecture, cost projections at scale — all already inscribed and citable. Happy to contribute drafting time.

What the grant could fund. Q1-Q4 above (API, UX, algorithm iteration, scaling). The pieces that need joint work but no single party has incentive to build alone.

Concrete first steps

Things we could do in the next 30 days:

STEP	EFFORT	WHAT IT SURFACES
Run the pipeline on 5 recent eLife papers of your choice	30 min wall time, ~$35	What the output looks like on papers your team picked, not ours
Wire claim files into a prototype editorial-surface	1 week (eLife UX side)	The claim-level review experience reviewers and editors would actually use
Pick one of Q1-Q4 to focus joint work	The conversation today	Where to invest first; what the partnership shape looks like
Draft the relevant section of the UK grant proposal	1-2 weeks coordinated	A concrete, fundable joint plan

Things to land in the next 90 days if the partnership proceeds:

- **HTTP API around the CLI** for editorial integration (~2 weeks)
- **Reference-set expansion** to 25-30 papers across more domains (~$200 + analyst time)
- **Targeted iteration** on the panel-assignment gap and the methods-paper variant
- **First-pass cross-paper UX** (claim browser, dependency viewer at corpus scale)

The system is operational today. The partnership shape is what we'd build together.

Documentation

For the depth behind these slides:

Docs site: `https://zmainen.github.io/elife-claim-trees/docs/`

SECTION	WHAT'S THERE
Overview	what this is / quick demo / when to use
Methodology	8-step / schema / curator-vs-system division
Architecture	pipeline / 3-agent / reconciliation / reviewer / review-gate
Using the CLI	install / first paper / subcommands / modes / batch / cost
Validation	methodology / 10-paper results / limitations / iteration
Reference	config / 9 roles / 14 edges / 5 prompts / API / glossary
Contributors	code structure / prompt variants / running validation / 10 ADRs

32 pages. Code-level when needed, narrative when concepts matter.

Source repo: *(link when published)*. Open to issues, PRs, prompt variants, anything.

Summary

The bigger arc	Move toward science where claims, not papers, are the unit of knowledge. Three layers: extraction (built), authoring (sketched), agent-driven (long horizon).
What's built	An eight-step extraction pipeline. Three Sonnet agents + Opus reconciliation + Opus structural-inference reviewer. CLI, validation harness, 32-page docs site.
Empirical performance	96.9% mean claim recovery, 83.1% mean role agreement, across 10 papers in the curated reference corpus. v2 prompts adopted as default.
What it costs	~$7 per paper with external reviewer; ~15 min per paper. Linear in corpus size; tractable to bioRxiv-scale with optimization (~$200-300K).
Where joint work matters	API + integration (Q1); UX surfaces (Q2); algorithm iteration (Q3); scaling engineering (Q4).
The shared opportunity	UK grant call featuring this kind of tech. Concrete, demonstrable, fundable. Convergence with OXA, elifePathways, the broader movement.

Q&A

Documentation: `https://zmainen.github.io/elife-claim-trees/docs/`

Source code: *(repo URL when published — open-sourced post-meeting)*

Contact: Zach Mainen, zmainen@neuro.fchampalimaud.org

Open questions for the conversation:

- Which of Q1-Q4 (API / UX / algorithm / scaling) is most urgent on your side?
- What's the right shape for editorial-system integration — HTTP API, file-based ingestion, modular publishing standard?
- What's the timeline on the UK grant — when would joint authorship of a section make sense?
- For elifePathways and OXA — who are the right people to bring into the next conversation?
- Where would a 5-paper proof-of-concept on eLife-team-chosen papers be most useful?

Pre-anticipated technical questions:

- *Could this run on non-eLife papers?* Yes, with adapted PDF fetch paths.
- *Could you fine-tune the underlying models?* Per-prompt variants today; weight fine-tuning is a future option if the reference corpus grows.
- *What about Step 8 verification?* Per-paper Python (`verify.py`); out of CLI scope. Existing verifiers in the corpus repo.
- *What about cross-paper graph traversal?* `verify-refs` provides DOI keys; entity resolution is a separate downstream system worth building.
- *What if a prompt change makes things worse?* The `evaluate` harness catches it; the change doesn't ship.