Agents at Your Service

Reuse-First Workflow Composition with Laminar 3.0

Marco Edoardo Santimaria, R. Filgueira, A. Mulone, I. Colonnelli, M. Aldinucci

21st Workshop on Workflows in Support of Large-Scale Science
Chicago, IL, USA

The Problem

Reuse is hard

Find, read, connect
All by hand

LLMs hallucinate

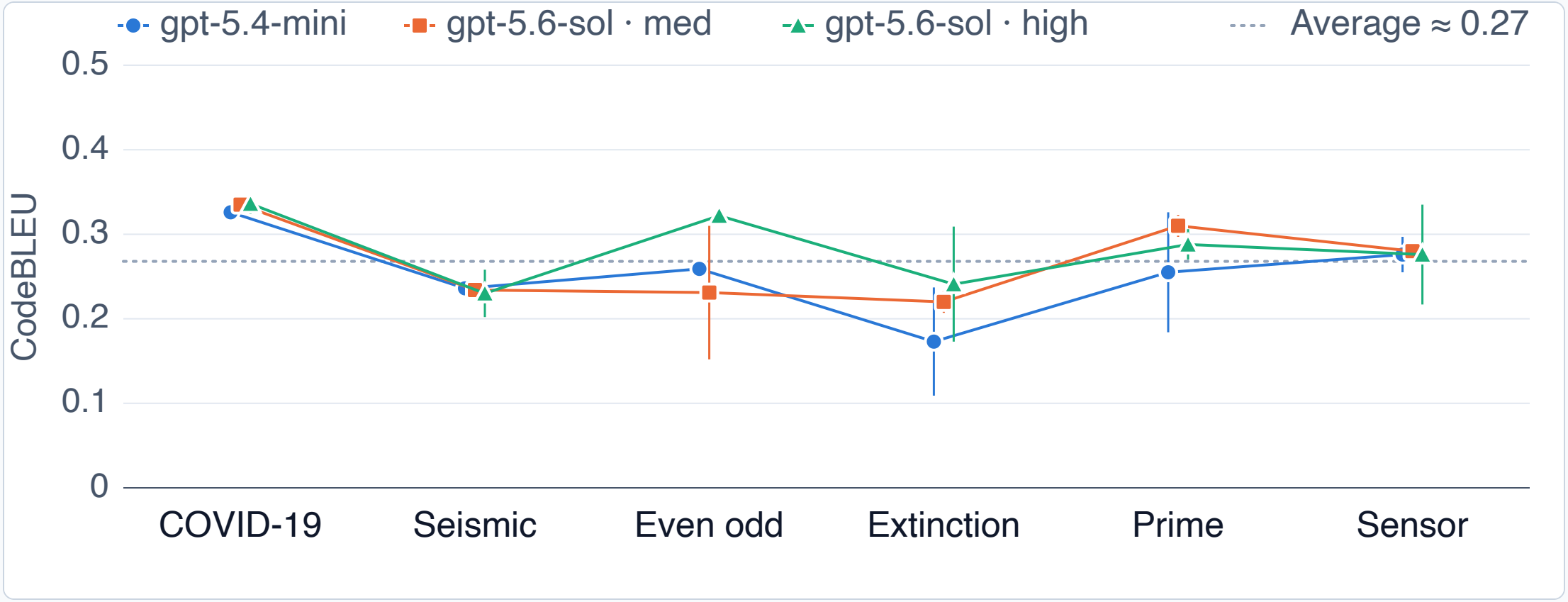
Invented components
and APIs

Control is lost

No traceability,
no review

Can LLMs compose workflows
while the developer stays in control?

Direct generation fails even with stronger models

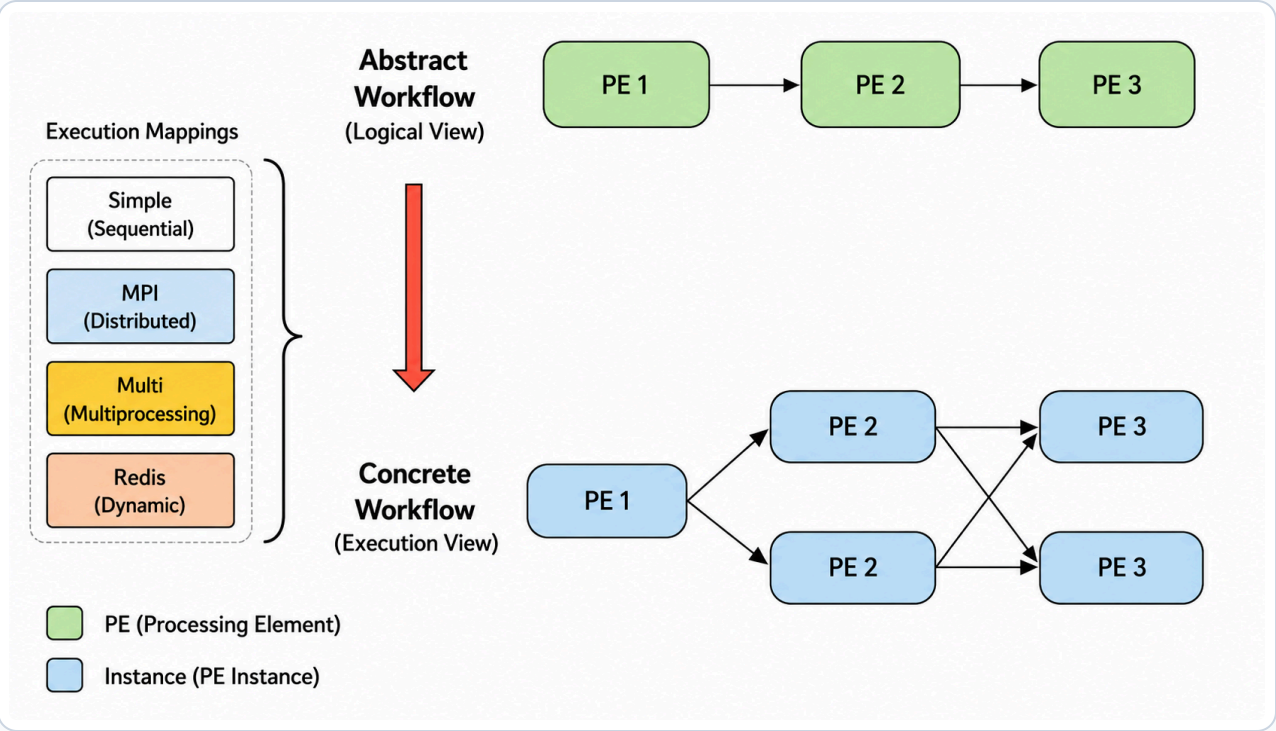


Note: **none of the generated workflows executed**

Background

Dispel4py

Dispel4py workflows are directed graphs of **processing elements** (PEs) joined by data streams between ports.



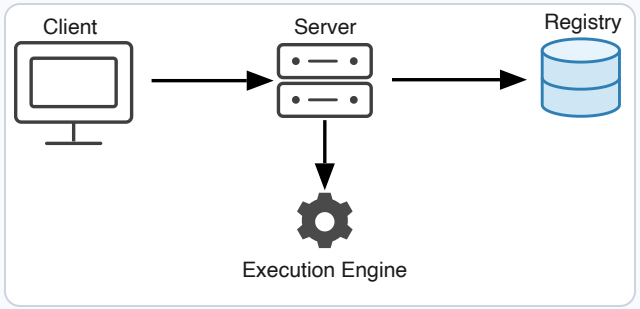
Abstract → concrete

An abstract graph runs sequentially or in parallel (MPI, Redis) with the same semantics.

This explicit representation is what Laminar registers, searches, and reuses.

Laminar: A Serverless Stream-based Framework

Every generation shares the same backbone.



1.0 — 2.0 — 3.0

Registry + execution

Serverless store & run; transformer search.

Python-first + CLI

Resource management human-led composition.

LLM × agents

Assist discovery *and* composition — human-in-the-loop.

> **Same architecture, growing capability** — only what assists the developer on top changes.

Approach

Laminar 3.0: retrieval-first, constraint-guided

Register & enrich

a description agent enriches every PE and workflow → registry + embeddings

↓

Search

classify → retrieve → re-rank · a strong match is reused as-is

↓ no strong match

Compose & validate

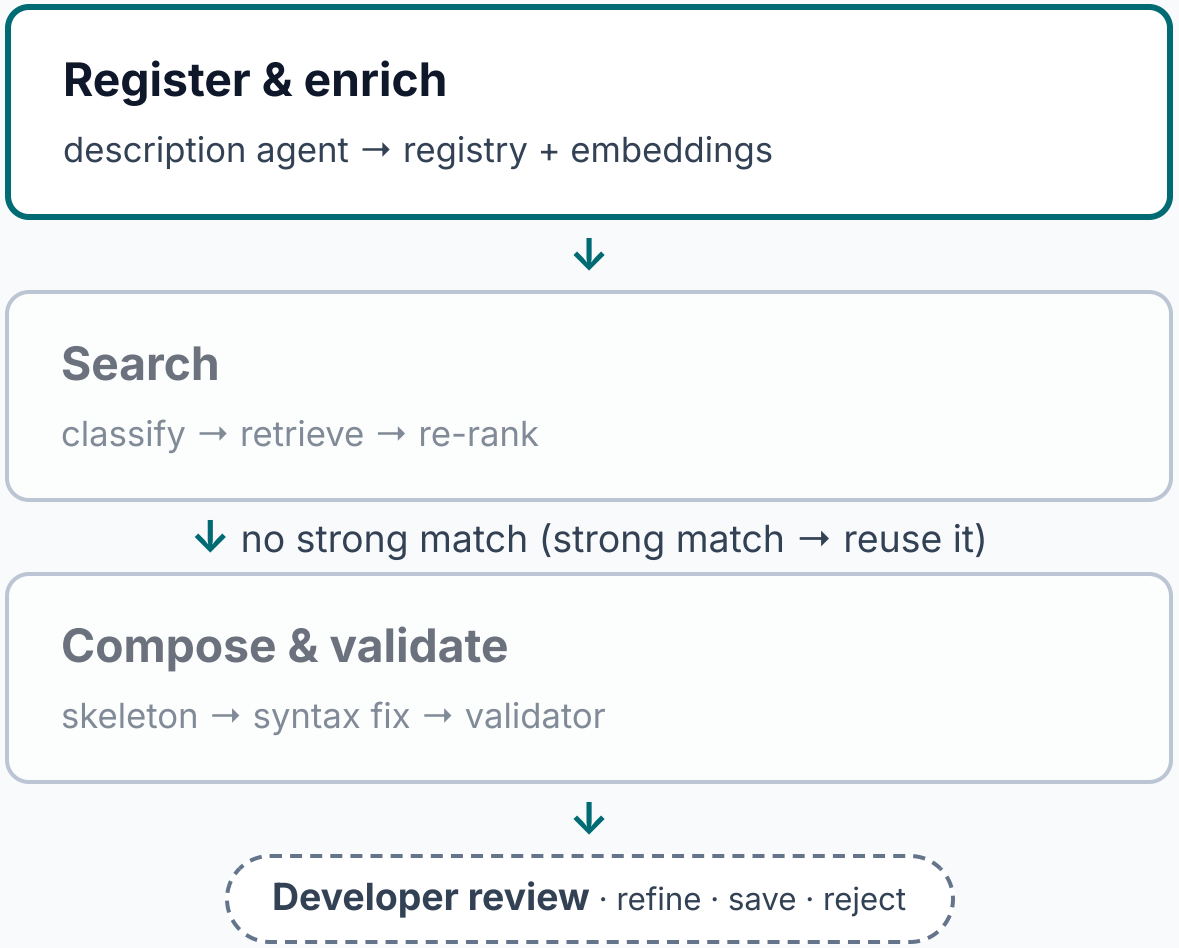
reuse retrieved PEs → draft a skeleton → syntax-check → validator

Reuse first

↓

Developer has the final say · refine · save · reject — never auto-registered or executed

Laminar 3.0 Registration and enrichment

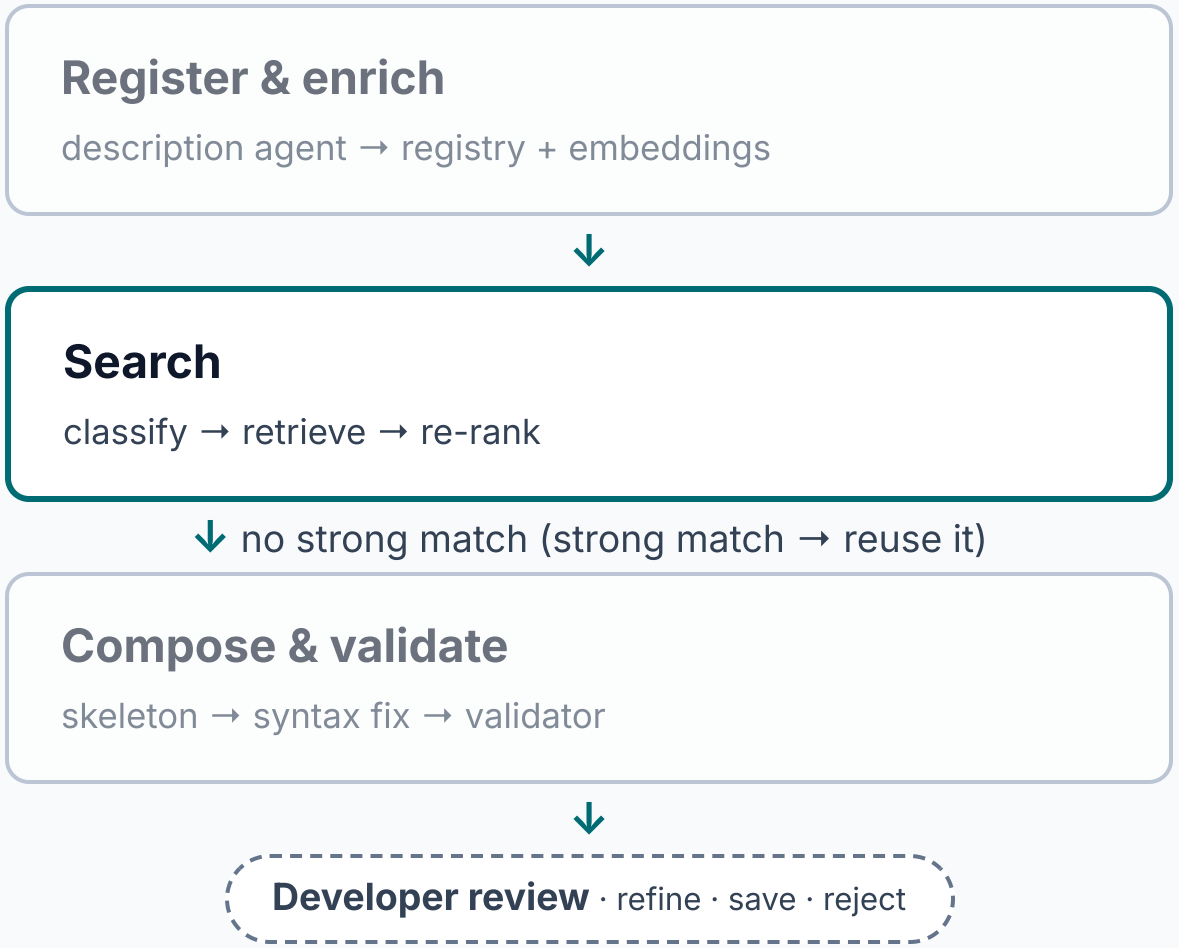


1 · Register & enrich

1. **Register** — the developer submits a PE or workflow: just source code, as in 2.0.
2. **Describe** — an LLM agent writes purpose, inputs/outputs, parameters, semantic tags; workflows also get per-step I/O.
3. **Index** — the record is stored in the registry and embedded for search.

> **vs. 2.0:** search matched raw code; now it matches *what a component does*.

Laminar 3.0 Advanced Search

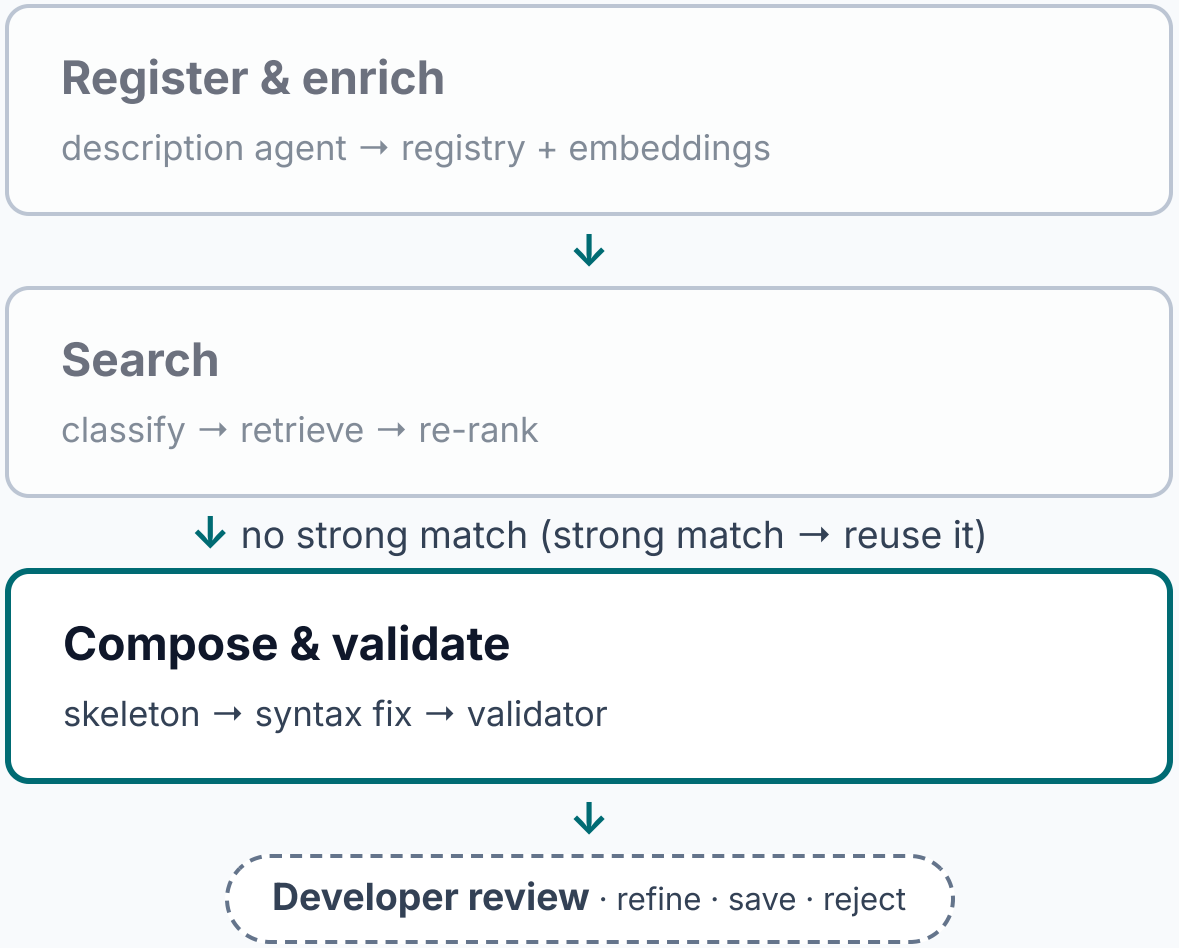


2 · Search the registry

1. **Classify** — an agent interprets the user request.
2. **Retrieve** — hybrid semantic + lexical search over the enriched records.
3. **Re-rank** — an LLM agent orders the candidates.
4. **Strong match?** — reuse it directly and present it to the developer.

> **Reuse first:** composition only starts when no component fits.

Laminar 3.0 Workflow Composition



3 · Compose & validate

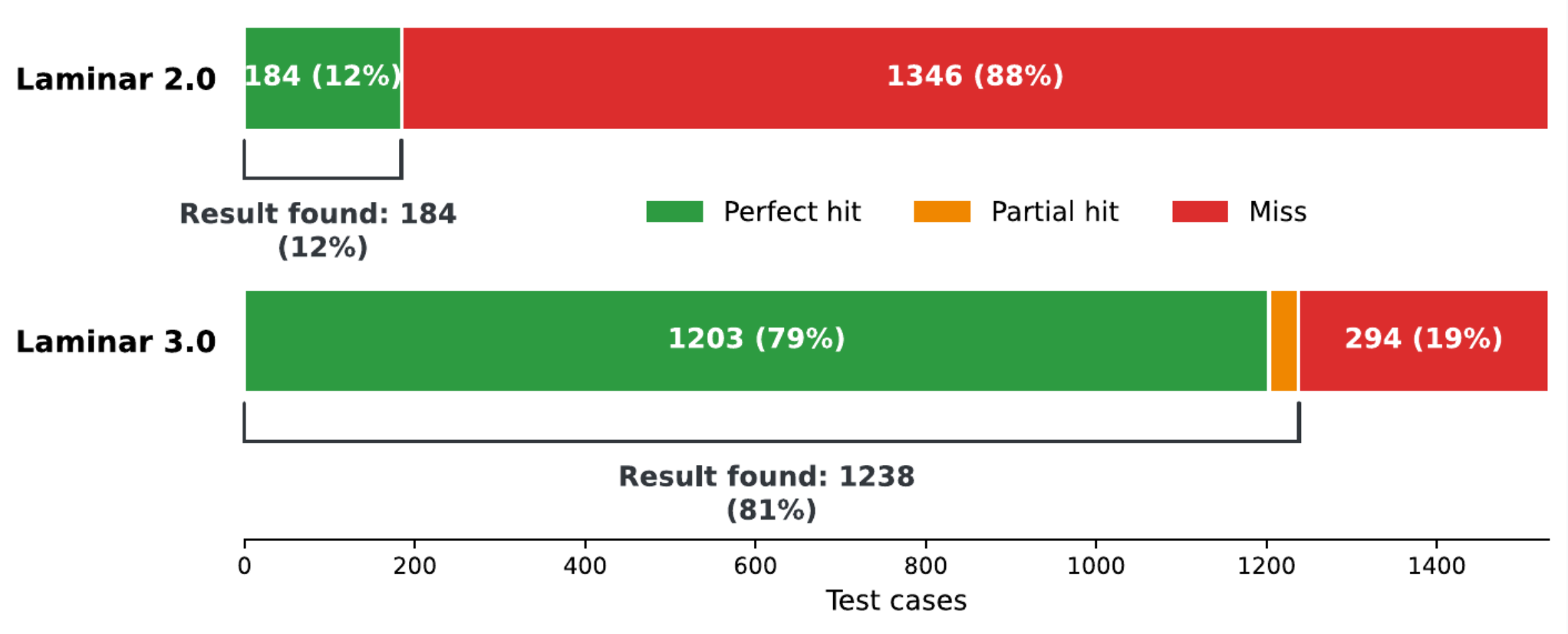
1. **Compose** — draft a skeleton under structured *hard-requirements* (step, workflow, format).
2. **Check & fix** — static syntax checks; a syntax agent repairs errors.
3. **Validate** — a strict validator agent; failures trigger regeneration.

→ The result goes to the developer to refine, save, or exit — never auto-executed.

> **Design:** LLMs handle the coordination language; developers keep the host logic.

Evaluation

Evaluation: retrieval is accurate



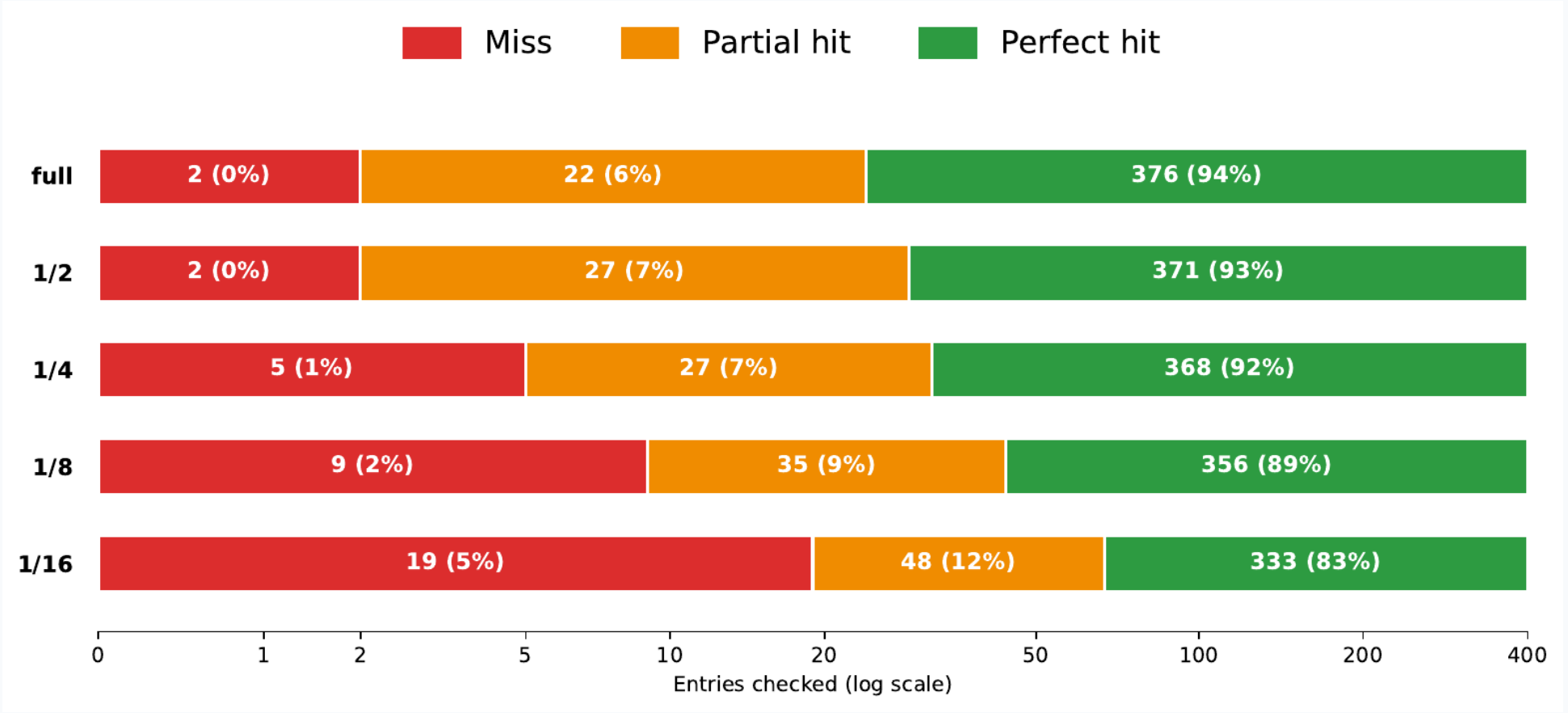
79%
top-1 perfect hit

81%
retrieved in top-3

6.75×
vs Laminar 2.0 (12%)

gpt-5.4-mini via Edinburgh's ELM · 8 workflows + 46 PEs, 1,530 authored queries.

Evaluation: robust to short queries



83%
top-1 at ~6 words

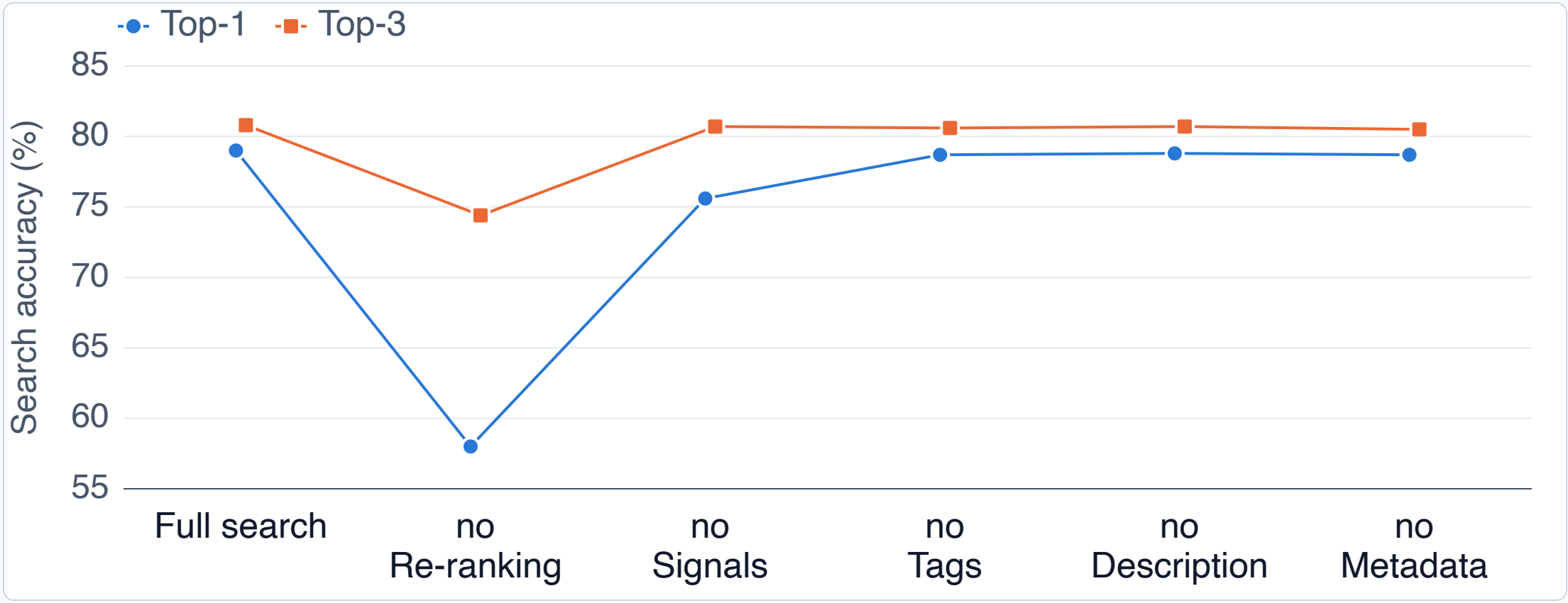
95%
within top-3 at ~6 words

2 → 19
misses as queries shrink

400 CodeSearchNet-derived PEs; queries compressed from ~100 down to ~6 words.

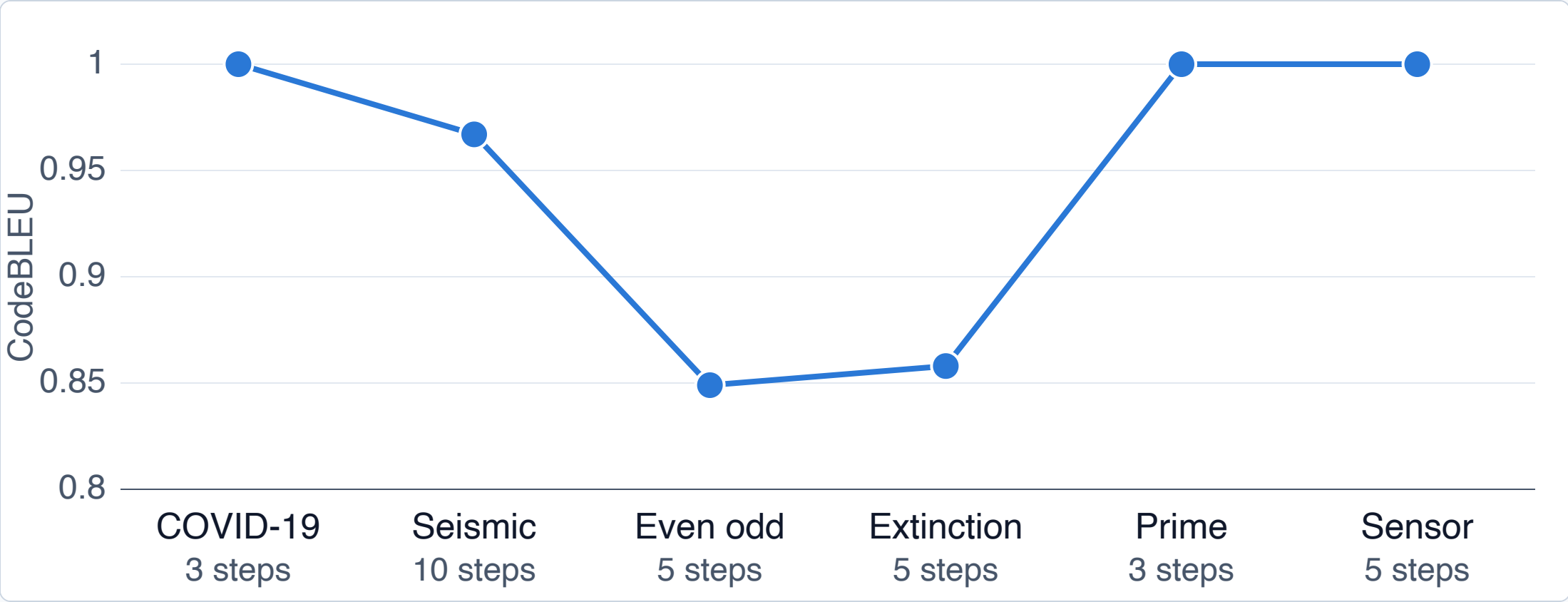
Evaluation: what drives retrieval

Ablation on the 1,530-query benchmark.



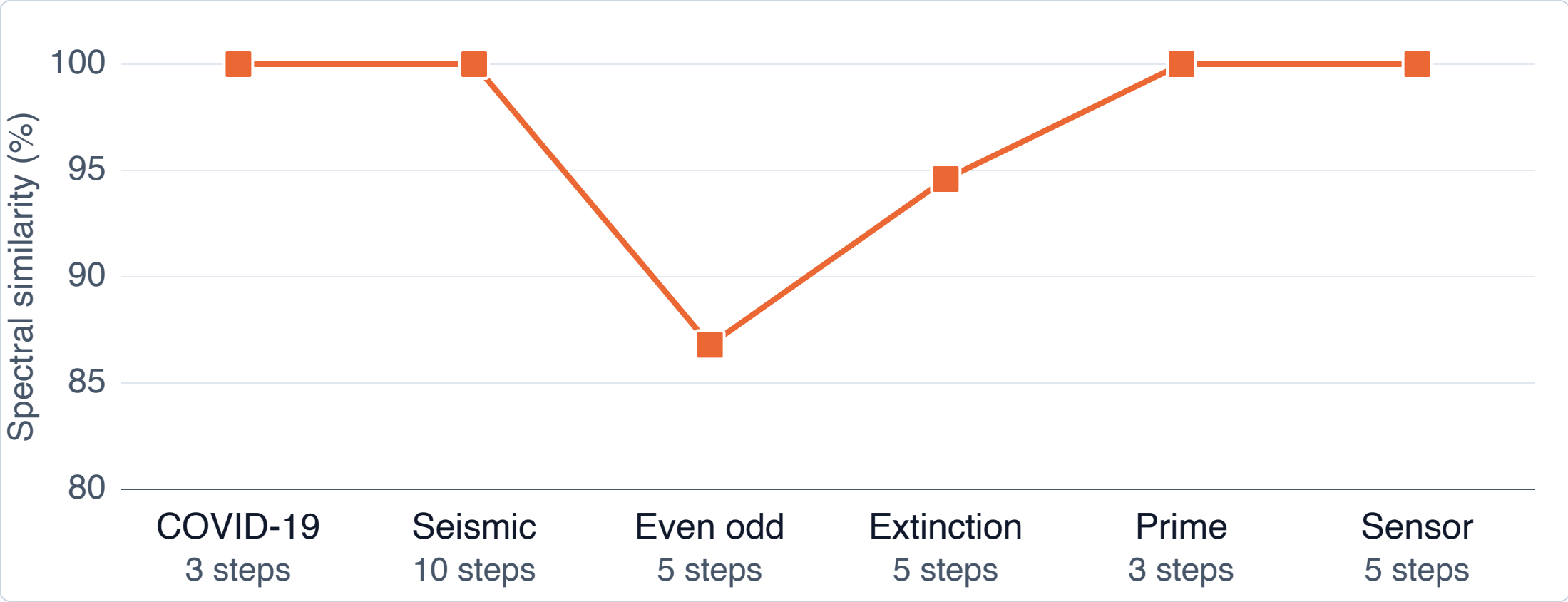
Evaluation: structural fidelity · CodeBLEU

Composed workflows vs. references, on the same 6 workflows as the direct-generation baseline.



Evaluation: structural fidelity · spectral similarity

Composed workflows vs. references, on the same 6 workflows as the direct-generation baseline.



Evaluation: structural fidelity · graph edit distance

Composed workflows vs. references, on the same 6 workflows as the direct-generation baseline.



UNIVERSITÀ DI TORINO

epcc

Conclusion

LLMs as **bounded agents**, not autonomous generators.

6.75× better retrieval than Laminar 2.0

3× CodeBLEU vs direct LLM generation

✔ **Human in control**: nothing runs without the developer

Next: agentic planning · SWIRL · PyCOMPSs/Nextflow

TRY IT ON GITHUB

Marco Edoardo Santimaria, R. Filgueira, A. Mulone, I. Colonnelli, M. Aldinucci

Thank you — questions?