POCKETLEAN: A LIGHTWEIGHT MATHLIB RETRIEVER FOR PROOF AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present PocketLean, a lightweight Mathlib retriever for proof agents that learns from their own retrieval calls. Agents query Mathlib, a large library that changes across versions, repeatedly as they write, check, and revise, so retrieval must be cheap to run and to re-index. Pairing each recorded call with the lemmas its Lean-checked final proof uses yields query-level weak supervision. After multi-source query training, R^2Evo (Replay-to-Rollout Evolution) applies this supervision over generations, first on recorded trajectories and then on rollouts collected with the updated retriever, so that the supervision follows the queries each update elicits. We also introduce ProofCall, a benchmark of 1,999 recorded agent calls in five sets, each labeled with the proof-used lemmas judged relevant to it, complementing the static queries of the public Lean Finder benchmark. Agents not told how to phrase their queries mostly issue approximate formal statements rather than natural-language questions, the form on which PocketLean is strongest. The 0.6B PocketLean reaches 78.9% mean Recall@10 on the Lean Finder benchmark against 77.9% for the 7B Lean Finder model, and the highest hit@10 on all five ProofCall sets, 4.3 to 7.6 points above the strongest external system. In end-to-end proving on subsets we curate from miniF2F and ProofNet, it more than doubles the success rate without retrieval, scores highest among the evaluated retrievers, and cuts the prover's tokens per model call by 36%. It needs only a single encoder and a per-version Mathlib index, and on a CPU it encodes 15 times as many queries per second as Lean Finder; we will release the weights, code, and data.

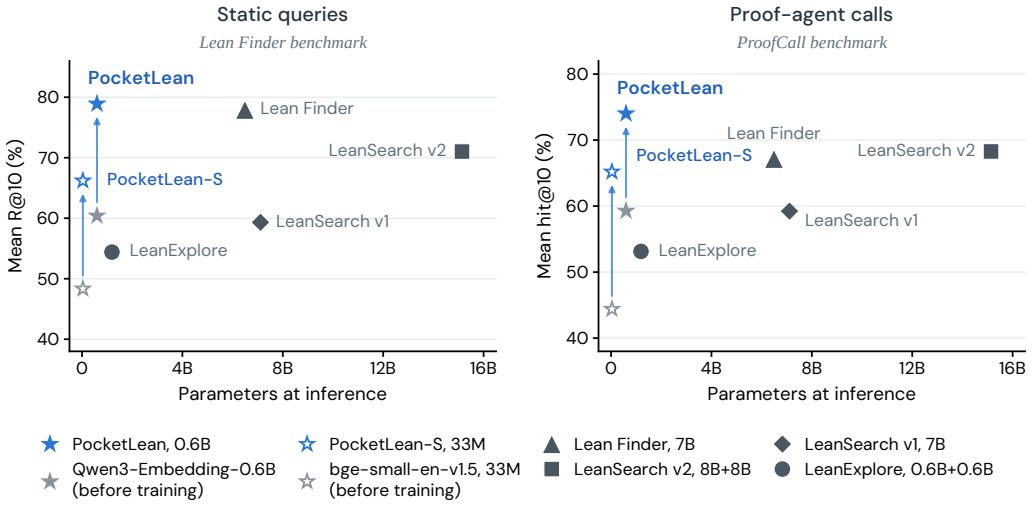

Figure 1: Retrieval quality versus parameters at inference. Scores are mean Recall@10 over the five query types of the Lean Finder benchmark (Lu et al., 2026) and mean hit@10 over the five sets of the ProofCall benchmark, the fraction of calls whose top ten include a proof-used lemma judged relevant to the call (Query labels). Arrows mark the gain from training each backbone.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

AI-driven mathematical formalization now reaches major results and large collaborative proof developments: Anthropic reports a formalization of Fermat's Last Theorem comprising approximately 13 million lines of Lean code (Anthropic, 2026a), and OpenAI has released results on the Navier–Stokes and Euler equations together with Lean formalizations (OpenAI, 2026c). These developments demonstrate agents' ability to construct complex formal proofs and highlight the need to organize and reuse existing mathematical knowledge. Such proofs are written in Lean (de Moura & Ullrich, 2021) and build on many existing definitions and lemmas (The mathlib Community, 2020). Mathlib holds more than a hundred thousand declarations that are renamed, added, and removed across versions; model parameters cannot reliably memorize a specific version, so proof agents depend on retrieval to use currently available results (Yang et al., 2023; Song et al., 2025). As models improve at mathematical reasoning and autoformalization, agents could contribute formalized results at greater scale (Chen et al., 2026), accelerating the growth and revision of libraries such as Mathlib. Repeated calls and frequent library updates therefore call for a lightweight retriever that rebuilds its index and answers queries at low cost.

Agents retrieve repeatedly as they write, check, and revise, with queries that mix natural-language descriptions, Lean identifiers, and unfinished proof drafts. One mathematical concept may also correspond to several Lean declarations with different assumptions and formulations, and the agent needs the one applicable to its current goal (Lu et al., 2026). Lean Finder, LeanSearch, and LeanExplore provide learned retrieval over Mathlib, and LeanStateSearch and REAL-Prover study proof-state retrieval (Lu et al., 2026; Gao et al., 2024; 2026; Asher, 2025; Tao et al., 2025; Shen et al., 2025). Static-query benchmarks, on which these systems are evaluated, do not establish retrieval quality on the requests agents actually issue. Recorded retrieval calls preserve agents' real requests, and Lean-checked final proofs reveal the lemmas actually used: replaying the calls gives a benchmark on real requests, and linking the two gives weak query-level supervision for retrieval training. LeanDojo and LeanAgent train retrievers from proof premises (Yang et al., 2023; Kumarappan et al., 2025); we attach proof dependencies to agents' recorded calls. Historical trajectories fix both queries and proofs and cannot show the follow-up queries an updated retriever would elicit, so we also collect new rollouts with the updated retriever.

We introduce PocketLean, a 0.6B Mathlib retriever trained in two stages: ***multi-source query training*** aligns the encoder with descriptions, formal fragments, proof states, and agent queries; ***R^{2}Evo (Replay-to-Rollout Evolution)*** then updates it with this proof supervision, first on recorded trajectories and then on new rollouts. At inference, the frozen encoder serves agents through a versioned Mathlib index (Figure 2). We evaluate it on the public Lean Finder benchmark and on ProofCall, a replay benchmark we build from the recorded calls. Our contributions are threefold:

1. **PocketLean, a Mathlib retriever that learns from proof agents' own retrieval calls.** We introduce PocketLean, a 0.6B Mathlib retriever trained on query-level weak supervision that pairs each retrieval call of a proof agent with the lemmas its Lean-checked final proof uses. Its training algorithm, R^{2}Evo, applies this supervision over generations, first on recorded trajectories and then on rollouts collected with each updated retriever, so that the supervision follows the queries the current retriever elicits (Sections 2.1 and 2.2).

2. **ProofCall, real agent calls as a retrieval benchmark.** We introduce ProofCall, which replays 1,999 recorded retrieval calls from trajectories not used for training, in five sets that differ in problems and collection conditions. Each call is labeled with the proof-used lemmas judged relevant to it. ProofCall complements the static queries of the public Lean Finder benchmark with the requests agents actually issue.

3. **A lightweight, high-performing retriever for the community.** The 0.6B PocketLean leads the strongest external system on every ProofCall set by 4.3 to 7.6 hit@10 points and exceeds the 7B Lean Finder model on the Lean Finder benchmark, 78.9% against 77.9% mean Recall@10. In end-to-end proving on subsets we curate from miniF2F and ProofNet (Zheng et al., 2022; Azerbayev et al., 2023), it more than doubles the success rate without retrieval and scores highest among the evaluated retrievers, while encoding queries on a CPU at practical speed. We will release the weights, code, and data.

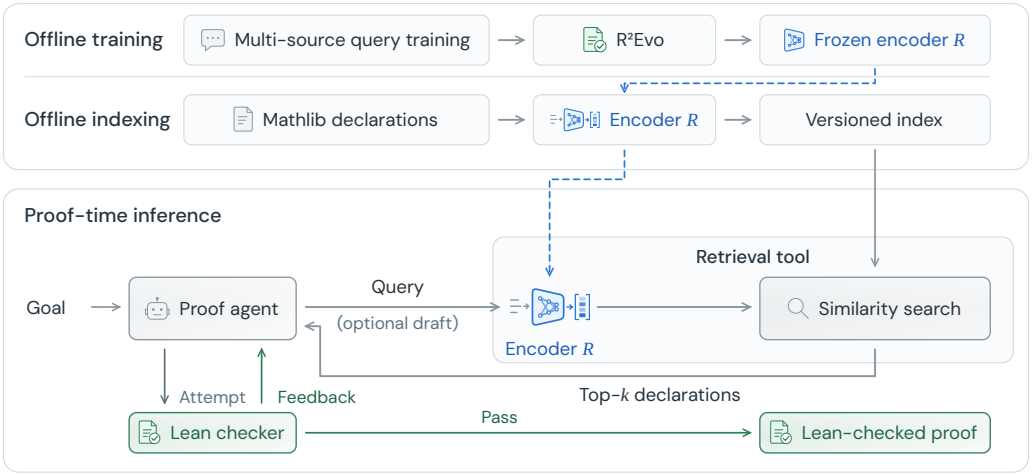

Figure 2: Training, indexing, and proof-time inference with PocketLean. PocketLean is the encoder R, trained as detailed in Figure 3 and then frozen: dashed arrows deploy its weights to encode Mathlib declarations and queries, and solid arrows show data flow.

2 POCKETLEAN

Task formulation. The retrieval input q combines an agent query with an optional existing proof draft, and may contain natural language, Lean identifiers, formal fragments, or proof states. For the agent's Mathlib version v, the retriever returns the top k candidates from the declaration corpus $\mathcal{C}_v$, $\mathrm{Retrieve}(q, \mathcal{C}_v, k) = (d_1, \ldots, d_k)$ with $d_i \in \mathcal{C}_v$. The current problem's final proof and its dependencies provide supervision retrospectively; they are not retrieval inputs.

Overview. PocketLean uses two training stages (Figure 3): multi-source query training initializes query–declaration matching, and R^{2}Evo then updates the retriever from proof experience. After training, the encoder is frozen and serves a proof agent through a per-version Mathlib index, as a single retrieval tool deployed separately from the proof-generation model (Figure 2).

2.1 MULTI-SOURCE QUERY TRAINING

Multi-source query training adapts a pretrained encoder to natural-language requests, formal fragments, and proof states before iterative updates from proof experience.

For declaration-derived examples, we pair descriptions, names, docstrings, corrupted formal statements, and synthetic questions with their source declarations. Proof-state examples pair LeanHammer states (Zhu et al., 2025) with their recorded premises. Following Lean Finder (Lu et al., 2026), we also generate intent descriptions from proof states and their next tactics; the next tactic is used only to construct the training query. Recorded agent queries provide additional query–lemma pairs from historical proof-use annotations.

A shared encoder e_θ scores queries and declarations by cosine similarity, $s_\theta(q, d) = \cos(e_\theta(q), e_\theta(d))$. Training combines annotated positives with highly ranked candidates mined outside each query's designated positive set. For a batch of B positive pairs (q_i, d_i^+), $C = (c_1, \ldots, c_M)$ concatenates batch positives and mined negatives; the contrastive loss (Karpukhin et al., 2020), with temperature $T > 0$, is

$$\mathcal{L}_{\mathrm{CL}}(\theta) = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(s_\theta(q_i, d_i^+)/T)}{\sum_{m=1}^{M} \exp(s_\theta(q_i, c_m)/T)}. \tag{1}$$

Training on these mixed sources initializes query–declaration matching across input forms, yielding parameters θ_0.

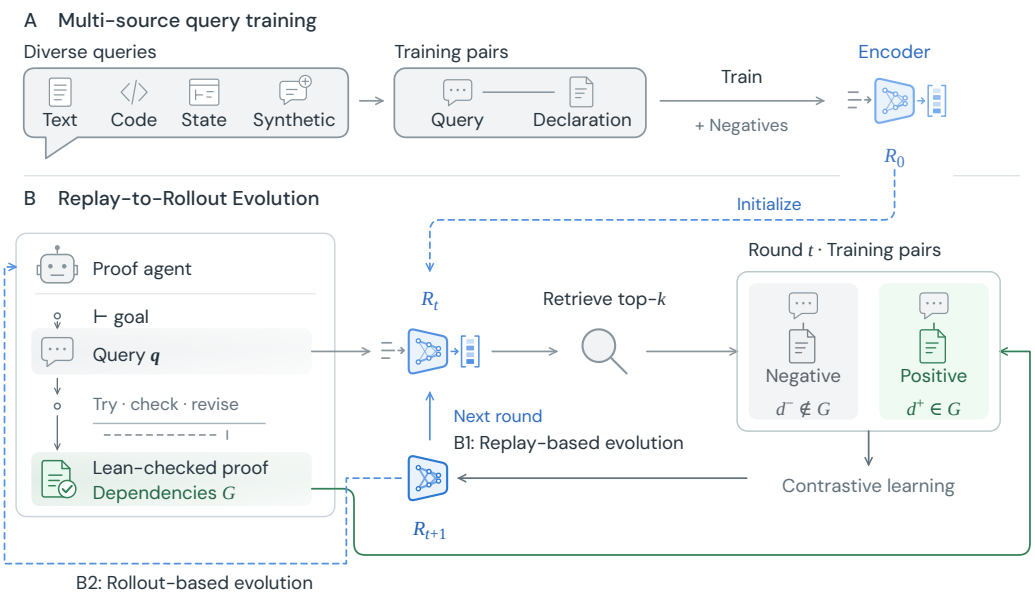

Figure 3: Training the PocketLean encoder: multi-source query training (A) and R^{2}Evo (B). Dependencies G of Lean-checked proofs provide weak labels for the training pairs. B1 replays recorded trajectories (solid loop); B2 also deploys each update to the agent to collect new ones (dashed loop). R_t is the encoder e_{θ_t} at round t; the last update gives PocketLean, the frozen R of Figure 2.

2.2 R^{2}Evo: Replay-to-Rollout Evolution

Starting from θ_0, R^{2}Evo constructs supervision from verified proof trajectories and refreshes hard negatives over successive updates. Replay-based evolution (B1) reuses fixed queries and proof paths shaped by earlier retrieval results, so it cannot observe the subsequent queries an updated retriever would elicit. Rollout-based evolution (B2) obtains this experience through new interactions with deployed retrievers.

Proof-derived supervision. For a trajectory ρ from either phase, let $q_1, \ldots, q_m$ be its ordered retrieval inputs and π_ρ its final proof; after rechecking in the matching Lean environment, we extract its proof-term dependencies. The eligibility set $\mathcal{E}_\rho$ retains indexed theorems named in the proof but not its statement, using full- or short-name heuristics and excluding tactic and utility modules; the labels are $G_\rho = \mathrm{Deps}(\pi_\rho) \cap \mathcal{E}_\rho$. An attribution rule $A_\rho(d)$ assigns call indices to each lemma, forming candidate positives:

$$P_\rho^{\mathrm{cand}} = \{(q_j, d) : d \in G_\rho,\ j \in A_\rho(d)\}. \tag{2}$$

These pairs remain weak supervision: proof verification does not certify query relevance, and G_ρ does not exhaust relevant declarations.

Each training group contains a query, one positive lemma, and negatives: top-ranked candidates of the parent encoder e_{θ_t}, from which the update continues training, that lie outside $\mathrm{Deps}(\pi_\rho)$ and share no short name with its members (Xiong et al., 2021).

B1: Replay-based evolution. Each update selects historical trajectories and assigns their proof-use lemmas to calls. **Trajectory-level attribution** assigns a lemma to every call of the trajectory. **Call-level attribution** assigns it to the earliest call that returned it while its name was absent from the pre-call context, the assistant text and selected tool arguments accumulated before that call. B1 uses either trajectory-level attribution or **call-level attribution with fallback**, which assigns a lemma without such a call to the last call before its name first appears, if any, to retain supervision for these lemmas. We continue training on new and retained groups and use the updated parameters.

B2: Rollout-based evolution. Each B2 update collects new rollouts with its parent retriever and the matching versioned index. B2 uses call-level attribution without the fallback, leaving lemmas without a qualifying call unassigned rather than attributing them by temporal proximity (Table 9 compares the three rules). To supervise retrieval in the local context of lemma use, we also replay verified proofs and pair pre-tactic states with lemmas named by that tactic and contained in G_ρ; the tactic itself is not an input. We continue training on new groups $\mathcal{D}_t$ mixed with retained groups $\mathcal{H}_t$, then average the continued model $\widetilde{\theta}_{t+1}$ equally with its parent θ_t, halving the update to temper adaptation to new interactions and retain existing retrieval ability (Wortsman et al., 2022a;b):

$$\begin{aligned} \widetilde{\theta}_{t+1} &= \mathrm{Train}(\theta_t, \mathrm{Mix}(\mathcal{D}_t, \mathcal{H}_t), \mathcal{L}_{\mathrm{CL}}), \\ \theta_{t+1} &= \tfrac{1}{2}\theta_t + \tfrac{1}{2}\widetilde{\theta}_{t+1}. \end{aligned} \tag{3}$$

2.3 Training data and the ProofCall benchmark

Training data. We denote the multi-source initialization by G0 (Section 2.1) and perform three replay updates (G1–G3) followed by two rollout updates (G4, G5); each update adds new proof-derived supervision to retained multi-source pairs. B1 replays historical trajectories on competition problems, and G3 adds trajectories on library theorems new in Mathlib 4.32, proved under 4.17; B2 collects rollouts in which more than a dozen policy models attempt library-theorem problems. PocketLean-S skips R^{2}Evo and is trained once from bge-small-en-v1.5 on the pooled training pairs (784,955 after deduplication). Appendix A lists the trajectories and pairs of each update (Table 8) with their sample construction and recipes.

ProofCall. To evaluate retrieval on agents' actual requests, we construct ProofCall, five replay sets of calls recorded during proof attempts on competition and library-theorem problems. No trajectory in the five sets is used for training. Each recorded call keeps its query, optional draft, and the trajectory's final proof. The call's labels are the lemmas of that proof that pass the dependency and text filters of Section 2.2, and a call paired with one of its labels is a query–lemma pair. Of the 3,083 calls, issued by proof agents built on more than a dozen policy models, the 2,489 whose final proof recompiled at construction time and whose label set is nonempty give 17,087 candidate pairs.

A proof's dependencies need not be relevant to every call that preceded it, so GPT-5.5 (OpenAI, 2026a) reviews each pair given the call, the lemma, and the proof context, without retriever names or ranks. It scores relevance from 0 (unrelated) to 3 (a direct answer), where 2 denotes a concrete intermediate fact useful for the query, and judges textual proof use separately. The reviews define three nested label sets: All, every pair; Proof, pairs judged to be used in the proof text; and Query, pairs judged relevant to the query with a score of at least 2. The Proof labels retain 95.0% of the pairs and the Query labels 34.3%; ProofCall scores retrievers against the Query labels on the 1,999 calls that retain one. Appendix B details ProofCall's construction, five sets, and label review.

3 Experiments

3.1 Experimental setup

Benchmarks. We evaluate in three settings: the Lean Finder benchmark, the ProofCall benchmark, and end-to-end proving on problem subsets we curate from miniF2F and ProofNet.

- **Lean Finder benchmark:** the public static benchmark of Lean Finder, with five query types and 7,448 queries evaluated against 244,521 candidate declarations (Lu et al., 2026).
- **ProofCall benchmark:** our replay benchmark of recorded proof-agent calls (Section 2.3), in five sets, Olympiad (4.32), Olympiad (4.17), Library-Theorem, Free-Form, and Form-Guided, with 203, 208, 302, 815, and 471 scored calls, each replayed against every retriever on a fixed corpus.
- **End-to-End Proving:** we evaluate proof success under a fixed tool-use budget on 37 miniF2F problems (Zheng et al., 2022) that Goedel-Prover-V2-32B (Lin et al., 2026) failed to prove in any of eight attempts (our *miniF2F-hard* subset), and on a curated 32-problem subset of ProofNet (Azerbayev et al., 2023); both subsets are constructed in this work (Appendix D).

Metrics. On the Lean Finder benchmark, Recall@k is the fraction of a query's labeled targets retrieved in the top k, averaged within each query type and then equally across the five types, following that benchmark's protocol (Lu et al., 2026). On ProofCall, hit@k is the fraction of scored calls whose top k results contain one of the call's labels, the Query labels of Section 2.3 unless stated otherwise; this adapts the premise-retrieval evaluation of LeanDojo (Yang et al., 2023) to recorded calls. End-to-end proving reports avg@8, the fraction of Lean-verified attempts among eight samples per problem and prover model, averaged over problems and prover models.

Models and configurations. PocketLean is trained from Qwen3-Embedding-0.6B (Zhang et al., 2025) and PocketLean-S from bge-small-en-v1.5 (Xiao et al., 2024); training settings are listed in Table 7. Baselines are the two off-the-shelf backbones, the Lean Finder paper checkpoint, and local adaptations of LeanSearch v1, LeanSearch v2 in standard mode, and LeanExplore (Lu et al., 2026; Gao et al., 2024; 2026; Asher, 2025) (Appendix E).

3.2 Main results

Table 1: Retrieval quality (%). Bold: best; underlined: second best; Avg.: ten-column mean; Aug./Inf.: augmented/informalized. *Reported by Lu et al. (2026).

		Lean Finder benchmark, R@10 ↑					ProofCall benchmark, hit@10 ↑					
Model	Params	Aug. stmt.	Aug. state	Inf. stmt.	User query	Raw state	Olymp. (4.32)	Olymp. (4.17)	Library thm.	Free-Form	Form-Guided	Avg. ↑
Off-the-shelf encoders												
bge-small-en-v1.5	33M	84.7	22.0	54.5	70.3	10.2	40.4	41.8	48.3	47.6	43.9	46.4
Qwen3-Embedding-0.6B	0.6B	82.3	33.0	80.1	84.9	21.7	74.9	55.8	53.3	61.0	51.4	59.8
Existing retrieval systems												
Lean Finder	7B	97.8	**70.5**	93.1	**92.7**	35.5	86.2	63.5	62.3	62.9	60.9	72.5
Lean Finder*	7B	97.7	67.9	93.3	91.4	40.0	–	–	–	–	–	–
LeanSearch v1	7B	89.3	28.2	77.3	81.7	20.1	66.5	58.2	55.0	60.7	55.8	59.3
LeanSearch v2	8B+8B	94.1	44.1	**94.6**	90.2	32.1	84.7	63.9	64.9	65.3	62.6	69.7
LeanExplore	0.6B+0.6B	81.1	29.3	68.9	74.1	18.6	53.2	51.4	52.6	55.6	52.9	53.8
PocketLean retrievers												
PocketLean-S	33M	98.7	44.6	74.6	81.0	32.1	79.3	63.0	56.6	63.8	63.3	65.7
△ vs. bge-small-en-v1.5		+14.0	+22.6	+20.1	+10.7	+21.9	+38.9	+21.2	+8.3	+16.2	+19.4	+19.3
PocketLean	0.6B	**99.4**	67.8	91.0	84.9	**51.5**	**90.6**	**69.7**	**69.9**	**69.6**	**70.3**	**76.5**
△ vs. Qwen3-Embedding-0.6B		+17.1	+34.8	+10.9	±0.0	+29.8	+15.7	+13.9	+16.6	+8.6	+18.9	+16.7

With one 0.6B dense encoder, PocketLean achieves the highest ten-column mean (76.5%; Table 1), 4.0 points above 7B Lean Finder. It leads all five ProofCall sets in hit@10 by 4.3–7.6 points over their strongest baselines, including LeanSearch v2 (8B encoder and 8B reranker). On the Lean Finder benchmark, mean R@10 reaches 78.9%, versus 77.9% (78.1% published) for Lean Finder. Relative to their backbones, PocketLean gains 8.6–18.9 hit@10 points on ProofCall, and 33M PocketLean-S gains 19.3 points overall. For PocketLean, the one Lean Finder benchmark type without a gain is User query (84.9% for both), whose 1,000 queries are all questions with a median length of 26 words. Among agent calls without query guidance, only 2.2% are user queries, while the most common type, augmented statements (62.9%), is PocketLean's best (99.4%; Appendix B.4).

3.3 Ablations and analysis

Retriever evolution across generations. Figure 4 follows one retriever through the five updates (per-set scores in Appendix A.4). On proof-agent calls, hit@10 rises at every update, from 66.4 to 74.0, and climbs most steeply in the first rollout round. Hit@1 falls once, at G1 (41.7 to 33.8), recovers at G2 (41.2), and then rises with every update to 46.9. The static curve stays nearly flat: Lean Finder Recall@10 remains within 0.7 points of G0 throughout and ends 0.7 points above it, so the gains on agent calls come without a loss in static retrieval. The two phases lift different sets: replay updates raise hit@10 on the two Olympiad sets and Library-Theorem by 4.4 points, and rollout rounds raise it on Free-Form and Form-Guided by 5.0 and 8.5; each phase gains most on calls similar to, but disjoint from, those that supply its supervision.

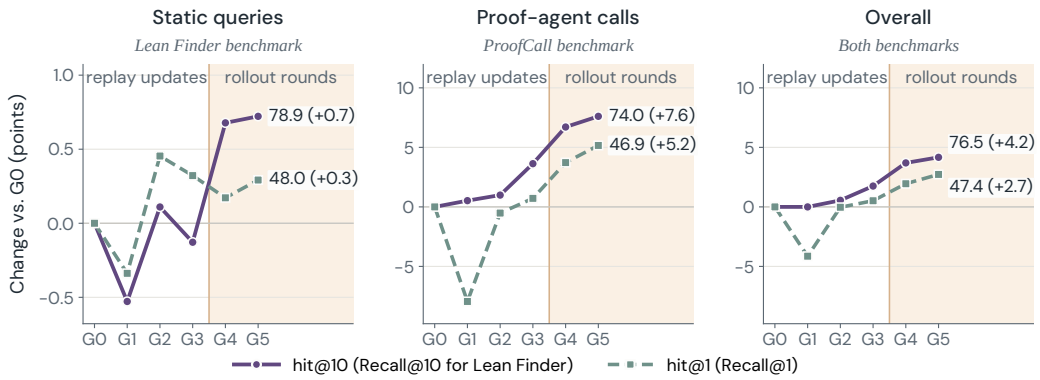

Figure 4: Change from G0 along the lineage, scored as in Figure 1; Overall is the ten-column mean (Avg. in Table 1). End labels give the value at G5 and, in parentheses, its change from G0.

Ablation study. Table 2 asks what each R^{2}Evo phase contributes and whether common inference components add to dense retrieval. Without B1, ProofCall hit@10 barely moves (−0.5) but hit@1 falls by 5.9 points, most on the two Olympiad sets (−14.8 and −8.2), whose calls on competition problems resemble, but do not overlap with, those B1 replays; B1 mainly brings a relevant lemma to the top. Without B2, hit@10 and hit@1 fall by 4.0 and 4.5 points; the hit@10 loss is largest on Free-Form and Form-Guided, whose calls from agents querying a deployed retriever resemble, but do not overlap with, the rollouts behind B2. Neither variant moves the Lean Finder benchmark by more than 1.0 point. Neither adaptive BM25 (Robertson & Zaragoza, 2009) nor reranking improves ProofCall hit@10, and hit@1 moves by at most 0.4 points. The reranker rescores only draft-free natural-language queries (Appendix E.2), so its gains on the Lean Finder benchmark fall on informalized statements and user queries, types that make up 18.5% of agent calls without query guidance (Appendix B.4). PocketLean therefore serves dense retrieval alone, with one encoder to deploy.

Precision of proof-derived supervision and its effect on training. Proof-derived supervision hinges on attribution: which earlier query a lemma used by the final proof should be paired with. On the three ProofCall sets that keep whole trajectories (456 trajectories, 1,982 calls with dependency labels), over 93% of the pairs selected by each attribution rule of Section 2.2 qualify as Proof labels, but the share that qualifies as Query labels, that is, relevant to the query, is 43.1% for trajectory-level attribution, 64.7% with fallback, and 87.5% for call-level attribution (Table 3). To test whether this difference matters for training, we continue G3 on the same retained data plus 30,287 pairs built from the same rollout trajectories under each rule, and compare against a control without these pairs. Continued directly, trajectory-level pairs lower ProofCall hit@1 by 4.2 points below the control,

Table 2: Ablations of PocketLean (%); columns as in Table 1. w/o B1 continues G0 on all B2 training data and averages it with G0; w/o B2 is G3. Small numbers: change from PocketLean.

	Lean Finder benchmark						ProofCall benchmark					
Variant	Aug. stmt.	Aug. state	Inf. stmt.	User query	Raw state	Mean	Olymp. (4.32)	Olymp. (4.17)	Library thm.	Free-Form	Form-Guided	Mean
R@10 and hit@10 ↑												
PocketLean	99.4	67.8	91.0	84.9	51.5	78.9	90.6	69.7	69.9	69.6	70.3	74.0
w/o B1	99.4 0.0	65.9 −1.9	91.0 0.0	84.8 −0.1	50.6 −0.9	78.3 −0.6	88.2 −2.4	67.8 −1.9	69.9 0.0	70.3 +0.7	71.5 +1.2	73.5 −0.5
w/o B2	99.3 −0.1	65.8 −2.0	91.0 0.0	85.2 +0.3	49.1 −2.4	78.1 −0.8	90.1 −0.5	66.8 −2.9	66.9 −3.0	64.5 −5.1	61.8 −8.5	70.0 −4.0
+ BM25	99.4 0.0	67.1 −0.7	91.0 0.0	85.2 +0.3	51.5 0.0	78.8 −0.1	90.1 −0.5	71.2 +1.5	69.2 −0.7	67.9 −1.7	70.5 +0.2	73.8 −0.2
+ reranker	99.4 0.0	67.8 0.0	93.6 +2.6	87.1 +2.2	51.5 0.0	79.9 +1.0	90.6 0.0	70.2 +0.5	69.9 0.0	69.0 −0.6	70.1 −0.2	73.9 −0.1
R@1 and hit@1 ↑												
PocketLean	88.1	25.6	60.2	51.6	14.5	48.0	75.4	46.2	38.1	37.8	36.9	46.9
w/o B1	88.1 0.0	23.9 −1.7	58.8 −1.4	49.3 −2.3	14.8 +0.3	47.0 −1.0	60.6 −14.8	38.0 −8.2	33.4 −4.7	36.4 −1.4	36.3 −0.6	41.0 −5.9
w/o B2	89.3 +1.2	25.5 −0.1	59.7 −0.5	51.0 −0.6	14.6 +0.1	48.0 0.0	72.9 −2.5	39.9 −6.3	32.5 −5.6	35.2 −2.6	31.6 −5.3	42.4 −4.5
+ BM25	90.1 +2.0	26.5 +0.9	60.1 −0.1	52.3 +0.7	14.5 0.0	48.7 +0.7	74.9 −0.5	46.6 +0.4	39.1 +1.0	39.1 +1.3	36.7 −0.2	47.3 +0.4
+ reranker	88.1 0.0	25.6 0.0	68.5 +8.3	55.3 +3.7	14.5 0.0	50.4 +2.4	75.4 0.0	45.7 −0.5	38.4 +0.3	36.8 −1.0	36.9 0.0	46.6 −0.3

significantly on the three replay sets, whereas call-level pairs raise hit@10 by 2.1 points. Averaged with G3 as in B2's update (Equation 3), trajectory-level pairs lose only 0.3 points of hit@1, all three rules raise Form-Guided hit@10 by 3.4 to 4.6 points, and call-level pairs keep the highest hit@1 (45.4 against 44.2 for trajectory-level pairs, significantly higher on Free-Form and Form-Guided). The Lean Finder benchmark changes by at most 0.7 points in every setting. These results support the two design choices of B2: call-level attribution puts more of the gain into ranking precision, and averaging each update with its parent keeps label noise from making the retriever worse.

Table 3: Retrieval quality (%) after continuing G3 with proof-derived pairs under each attribution rule; ProofCall hit and Lean Finder R as in Table 1, means over two seeds (per set: Table 10). Query: share of selected pairs qualifying as Query labels. Small numbers: change from the control.

			Continued model				Averaged with G3 (Eq. 3)			
Supervision added	Used by	Query (%)	hit@10	hit@1	R@10	R@1	hit@10	hit@1	R@10	R@1
None (control)		–	72.1	44.8	78.2	48.0	71.1	44.4	78.4	48.1
Trajectory-level	G1, G2	43.1	71.6 −0.5	40.6 −4.2	77.6 −0.6	47.3 −0.7	71.9 +0.8	44.2 −0.2	78.3 −0.1	47.9 −0.2
Call-level with fallback	G3	64.7	73.6 +1.5	43.6 −1.2	77.8 −0.4	47.2 −0.8	72.6 +1.5	45.1 +0.7	78.3 −0.1	47.7 −0.4
Call-level	G4, G5 (adopted)	87.5	74.1 +2.0	46.0 +1.2	78.2 0.0	47.7 −0.3	72.5 +1.4	45.4 +1.0	78.4 0.0	48.2 +0.1

Sensitivity to label review. Figure 5 rescores the cached top-10 rankings of all eight retrievers on the same 1,999 calls under the three label sets, with per-set scores in Appendix B.3. Under All and Proof labels PocketLean averages 77.1% and 76.5% hit@10 against 74.0% under Query labels, and it keeps the highest score on every set under all three label sets. Its lead over LeanSearch v2 narrows from 7.9 to 5.7 points from the All labels to the Query labels. On all 2,489 calls with dependency labels PocketLean still leads at 66.3%, and PocketLean-S exceeds Lean Finder, 58.1% against 57.3% (Table 14).

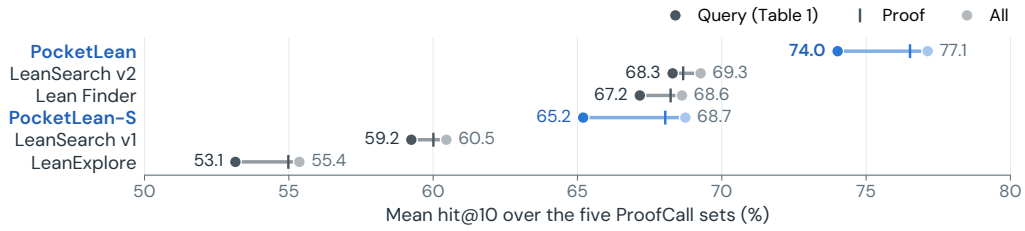

Figure 5: Mean hit@10 over the five ProofCall sets on the same 1,999 calls, rescoring each retriever's fixed top-10 rankings under the three label sets of Section 2.3; Table 1 uses the Query labels. The two off-the-shelf encoders are omitted; Table 15 lists all eight retrievers.

3.4 End-to-end proving and deployment

End-to-end proving. Table 4 reports end-to-end proving on miniF2F-hard (37 problems) and ProofNet (32) with five prover models and eight samples per condition: the agent may query the retriever up to four times and then submits one proof, without compiler feedback. All retrieval conditions share the same corpus and service configuration and differ only in the encoder (Appendix D). PocketLean more than doubles avg@8, adding 5.4 points over no retrieval. With 0.6B parameters it also leads Lean Finder (7B) and LeanSearch v2 (8B) by 1.3 and 1.1 points, and has the lowest tokens per call among the retrievers.

Quality against size and speed. Table 5 compares both PocketLean models with the two 7B retrievers on one machine (Appendix F). PocketLean exceeds Lean Finder in quality with one twelfth of its parameters and 14.6 times its CPU throughput, and the 33M PocketLean-S encodes 411 queries per second on the CPU while still exceeding the 7B LeanSearch v1 in quality.

Table 4: End-to-end proving on miniF2F-hard and ProofNet, one submission per attempt. avg@8: fraction of eight samples proved; tokens per call: tokens per model call, including thinking; vs. none: relative change over no retrieval. Best in bold, second best underlined.

		avg@8 (%) ↑				Tokens / call (k) ↓	
Model	Params	miniF2F-hard	ProofNet	Both	vs. none	Mean	vs. none
No retrieval	–	0.4	6.9	3.4	–	18.4	–
Lean Finder	7B	1.8	14.1	7.5	+119.1%	12.4	-32.7%
LeanSearch v1	7B	1.6	13.2	7.0	+104.9%	12.2	-33.6%
LeanSearch v2	8B	<u>1.9</u>	<u>14.4</u>	<u>7.7</u>	<u>+125.5%</u>	<u>12.0</u>	<u>-34.9%</u>
LeanExplore	0.6B+0.6B	1.4	13.0	6.8	+99.1%	12.5	-32.0%
PocketLean	0.6B	**3.0**	**15.5**	**8.8**	**+157.4%**	**11.8**	**-36.0%**

Table 5: Query-encoding throughput on the same 400 queries (GPU: H100; CPU: 16-thread Xeon; bf16, batch 16), peak resident memory on the CPU (with 2.3 GB of runtime), and the ten-column mean of Table 1. vs. Lean Finder: ratios, or point gaps for quality. Reranking systems are omitted.

System	Params ↓	Peak RSS ↓	GPU queries/s ↑	CPU queries/s ↑	Quality ↑
Lean Finder	7B	18.2 GB	256	3.6	72.5
LeanSearch v1	7B	21.2 GB	211	2.9	59.3
PocketLean	0.6B	5.3 GB	437	52.8	**76.5**
vs. Lean Finder	1/12	0.29×	1.7×	14.6×	+4.0
PocketLean-S	33M	2.3 GB	1,402	410.6	65.7
vs. Lean Finder	1/212	0.13×	5.5×	113.8×	−6.8

4 Related work

Mathlib retrieval and premise selection. Lean Finder trains on queries built from user intents in public Lean discussions, LeanSearch uses general-purpose embedding models with a reranker added in v2, and LeanExplore combines lexical search, embedding similarity, and dependency-graph scores (Lu et al., 2026; Gao et al., 2024; 2026; Asher, 2025; 2026). Herald annotates the Lean 4 mathematical library in natural language, and doc2query synthesizes queries from documents (Gao et al., 2025; Nogueira et al., 2019). Premise selection retrieves lemmas for proof states in Isabelle and Lean (Mikuła et al., 2024; Yang et al., 2023; Piotrowski et al., 2023; Zhu et al., 2025; Tao et al., 2025). CSLibPremiseBench also studies label robustness in premise retrieval, with source-visible references rather than elaborated proof dependencies as labels (Ji, 2026). **Retriever training from proofs.** REAL-Prover trains a retriever on proof states and used premises with mined hard negatives, and its expert iteration updates the prover (Shen et al., 2025). LeanAgent trains a retriever along a curriculum of mathematical repositories for lifelong learning (Kumarappan et al., 2025). LRAT learns retrievers from search-agent trajectories, constructing and weighting supervision from browsing choices and subsequent reasoning (Zhou et al., 2026). R^2Evo instead takes weak labels from dependencies of Lean-checked final proofs. Its rollout-based evolution alternates search and learning as in expert iteration, but updates the retriever rather than the prover (Anthony et al., 2017; Polu et al., 2023).

5 Conclusion

We introduced PocketLean, a Mathlib retriever that learns from proof agents' own retrieval calls, each paired with the lemmas its Lean-checked final proof uses. R^2Evo extends this supervision from recorded trajectories to rollouts collected with the updated retriever, which bring the largest gains on calls from deployed retrievers. ProofCall turns 1,999 such calls into a benchmark. Agent calls consist mostly of approximate formal statements rather than natural-language questions, and reranking, which helps on static queries, does not raise hit@10 on them, so PocketLean needs only one encoder. The 0.6B PocketLean leads on every ProofCall set and in mean Recall@10 on the Lean Finder benchmark, ahead of retrievers more than ten times its size; in end-to-end proving, it more than doubles the success rate without retrieval and cuts the prover's tokens per call by more than a third.

AI USE STATEMENT

LLMs are part of the method and data pipeline described in this paper: they generate declaration descriptions, synthetic questions, and proof-state intents for training (Appendix A), act as the proof agents that produce the recorded trajectories, and review ProofCall labels (Section 2.3). The authors designed the core ideas and architecture of the experimental code, and AI coding assistants helped complete and refine the implementation; AI tools also helped write plotting code and edit the text. The research questions and method design are the authors'; the authors reviewed all AI-assisted code, data, and text and take responsibility for the final content.

REFERENCES

Alibaba Cloud. Qwen3.8-Flash released, August 2026a. URL https://www.alibabacloud.com/en/news/product/qwen38-flash-released-noi.

Alibaba Cloud. Alibaba unveils Qwen3.8-Max: Its largest and most capable flagship model to date, August 2026b. URL https://www.alibabacloud.com/en/press-room/alibaba-unveils-qwen3-8-max.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Anthropic. Formalizing Fermat's Last Theorem. https://www.anthropic.com/research/formalizing-fermats-last-theorem, September 2026a. Research report, September 4.

Anthropic. Introducing Claude Opus 5, July 2026b. URL https://www.anthropic.com/news/claude-opus-5.

Justin Asher. Leanexplore: A search engine for lean 4 declarations. *arXiv preprint arXiv:2506.11085*, 2025.

Justin Asher. LeanExplore: Local search backend. https://github.com/justincasher/lean-explore/blob/6b25f8632cc3387cf85f7730375a690a1f1dfb79/docs/local-backend.md, 2026. Implementation documentation, accessed September 9, 2026.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023. URL https://arxiv.org/abs/2302.12433.

Shuze Chen, Kunal Marwaha, Xiaoyang Lu, Henry Yuen, and Tianyi Peng. Prove2Me: An open collaborative platform for scaling math formalization. *arXiv preprint arXiv:2608.28433*, 2026.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759, 2009. URL https://doi.org/10.1145/1571941.1572114.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, 2021.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Guoxiong Gao, Haocheng Ju, Jiedong Jiang, Zihan Qin, and Bin Dong. A semantic search engine for mathlib4. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. Herald: A natural language annotated lean 4 dataset. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/8c2bb821410066459be64d03a4dc5719-Abstract-Conference.html.

Guoxiong Gao, Zeming Sun, Jiedong Jiang, Yutong Wang, Jingda Xu, Peihao Wu, Bryan Dai, and Bin Dong. Leansearch v2: Global premise retrieval for lean 4 theorem proving. *arXiv preprint arXiv:2605.13137*, 2026.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP)*, 2021.

Google DeepMind. Gemini 3.7 Flash model card, August 2026. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/.

Junye Ji. CSLibPremiseBench: Structure-guided premise retrieval and label robustness for lean 4 computer-science theorems. *arXiv preprint arXiv:2605.14549*, 2026. URL https://arxiv.org/abs/2605.14549.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*, 2020.

Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. URL https://arxiv.org/abs/2607.24653.

Adarsh Kumarappan, Mo Tiwari, Peiyang Song, Robert Joseph George, Chaowei Xiao, and Anima Anandkumar. LeanAgent: Lifelong learning for formal theorem proving. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=Uo4EHT4ZZ8.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover-V2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2508.03613.

Jialin Lu, Kye Emond, Kaiyu Yang, Swarat Chaudhuri, Weiran Sun, and Wuyang Chen. Lean finder: Semantic search for mathlib that understands user intents. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=5XNnnbEcu5. arXiv:2510.15940.

Yuanhua Lv and Chengxiang Zhai. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pp. 7–16, 2011. doi: 10.1145/2063576.2063584.

Maciej Mikuła, Szymon Tworkowski, Szymon Antoniak, Bartosz Piotrowski, Albert Q. Jiang, Jin Peng Zhou, Christian Szegedy, Łukasz Kuciński, Piotr Miłoś, and Yuhuai Wu. Magnushammer: A transformer-based approach to premise selection. In *International Conference on Learning Representations (ICLR)*, 2024.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*, 2019.

OpenAI. GPT-5.5 system card, April 2026a. URL https://openai.com/index/gpt-5-5-system-card/.

OpenAI. GPT-5.6 Luna. Model documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna. Accessed September 26, 2026.

OpenAI. On the Navier–Stokes millennium prize problem. https://openai.com/index/navier-stokes-solution/, September 2026c. Research report, September 8; accompanying Lean formalizations of Navier–Stokes and Euler results.

Bartosz Piotrowski, Ramon Fernández Mir, and Edward Ayers. Machine-learned premise selection for lean. *arXiv preprint arXiv:2304.00994*, 2023. URL https://arxiv.org/abs/2304.00994.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=-P7G-8dmSh4.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Ziju Shen, Naohao Huang, Fanyi Yang, Yutong Wang, Guoxiong Gao, Tianyi Xu, Jiedong Jiang, Wanyi He, Pu Yang, Mengzhou Sun, Haocheng Ju, Peihao Wu, Bryan Dai, and Bin Dong. REAL-Prover: Retrieval augmented lean prover for mathematical reasoning. *arXiv preprint arXiv:2505.20613*, 2025. URL https://arxiv.org/abs/2505.20613.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Lean Copilot: Large language models as copilots for theorem proving in Lean. In *Proceedings of the International Conference on Neuro-symbolic Systems*, volume 288 of *Proceedings of Machine Learning Research*, pp. 144–169. PMLR, 2025. URL https://proceedings.mlr.press/v288/song25a.html.

Yicheng Tao, Haotian Liu, Shanwen Wang, and Hongteng Xu. Learning an effective premise retrieval model for efficient mathematical formalization. *arXiv preprint arXiv:2501.13959*, 2025. URL https://arxiv.org/abs/2501.13959.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pp. 367–381, 2020. doi: 10.1145/3372885.3373824. URL https://arxiv.org/abs/1910.09336.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11897–11916, 2024. doi: 10.18653/v1/2024.acl-long.642. URL https://aclanthology.org/2024.acl-long.642/.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022a.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7959–7971, 2022b.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. URL https://arxiv.org/abs/2309.07597.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Haowei Zhang, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo tree search. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/b3b55c366d641c07180c40e4f978f311-Abstract-Conference.html.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations (ICLR)*, 2021.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Z.ai. GLM-5.2: Built for long-horizon tasks, June 2026. URL https://z.ai/blog/glm-5.2.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: A cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

Yuqi Zhou, Sunhao Dai, Changle Qu, Liang Pang, Jun Xu, and Ji-Rong Wen. Learning to retrieve from agent trajectories. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2026. URL https://arxiv.org/abs/2604.04949.

Thomas Zhu, Joshua Clune, Jeremy Avigad, Albert Qiaochu Jiang, and Sean Welleck. Premise selection for a lean hammer. *arXiv preprint arXiv:2506.07477*, 2025.

A TRAINING SOURCES AND CHECKPOINTS

Multi-source query training runs as an initial pass (Init) and a continuation on a subsample that yields G0 (Table 6); Algorithm 1 summarizes the full training procedure of the main model.

Algorithm 1 Multi-source training and R^{2}Evo

Require: θ_{pre}, $\mathcal{D}_{\text{multi}}$, versioned corpora, update configurations
Require: Historical trajectories, training problems, retained groups $\mathcal{H}_t$
Require: B1/B2 update counts K_1, K_2 and candidate budgets b_1, b_2
Ensure: Retriever checkpoints after each update

$\theta_0 \leftarrow \textsc{Train}(\theta_{\text{pre}}, \mathcal{D}_{\text{multi}}, \mathcal{L}_{\text{CL}})$ ▷ A: Initialization
for $t = 0, \ldots, K_1 - 1$ **do**
  $\mathcal{T}_t \leftarrow \textsc{SampleHistorical}(\text{update configuration})$
  $\textsc{RefreshIndex}(\theta_t)$
  $\mathcal{D}_t \leftarrow \textsc{BuildCalls}(\mathcal{T}_t, \theta_t, \text{B1 attribution}, b_1)$
  $\theta_{t+1} \leftarrow \textsc{Train}(\theta_t, \text{Mix}(\mathcal{D}_t, \mathcal{H}_t), \mathcal{L}_{\text{CL}})$
  Save θ_{t+1} ▷ B1: Replay update
end for
for $t = K_1, \ldots, K_1 + K_2 - 1$ **do**
  $\textsc{RefreshIndex}(\theta_t)$
  $\mathcal{T}_t \leftarrow \textsc{Acquire}(\text{agent rollouts with } \theta_t)$
  $\mathcal{D}_t \leftarrow \textsc{BuildCalls}(\mathcal{T}_t, \theta_t, \text{call-level attribution}, b_2)$
  $\mathcal{D}_t \leftarrow \text{Concat}(\mathcal{D}_t, \textsc{BuildStates}(\mathcal{T}_t, \theta_t, b_2))$
  $\widetilde{\theta}_{t+1} \leftarrow \textsc{Train}(\theta_t, \text{Mix}(\mathcal{D}_t, \mathcal{H}_t), \mathcal{L}_{\text{CL}})$
  $\theta_{t+1} \leftarrow (\theta_t + \widetilde{\theta}_{t+1})/2$
  Save θ_{t+1} ▷ B2: Rollout update
end for

BUILDCALLS and BUILDSTATES recheck proofs, extract G, construct weak supervision, filter negatives, and sample groups; provenance follows Appendix A.3 and budgets Table 7. ACQUIRE uses a matching index for each deployed encoder; TRAIN uses the update budget under Equation 1 and does not update proof-agent parameters.

Table 6: Multi-source training pairs by source for the initial pass (Init) and the continuation yielding G0.

Query source	Init	G0
Released declaration descriptions	239,606	60,000
Short declaration names	120,000	20,000
Descriptions from three annotated versions	317,039	60,000
Corrupted formal statements	199,993	40,000
Synthetic user questions	121,121	60,000
LeanHammer proof states	400,000	110,000
State intents	131,111	50,000
Recorded agent queries	24,319	24,320
Total	1,553,189	424,320

Table 7: Training configurations; mini-batch is the gradient-cache chunk size. All models are fully fine-tuned with normalized embeddings, a 512-token input limit, and Equation 1 at temperature 0.05, gathering in-batch negatives across devices in multi-GPU runs. Replay and rollout updates use candidate budgets of 20 and 10; G4 and G5 are averaged with their parents by Equation 3.

Checkpoint	Initialization	GPUs	Batch/device	Mini-batch	Epochs	LR	Warmup
Init	Base Qwen	6	128	16	1	2×10^{-5}	5%
G0–G5	Parent	1	288	8	1	8×10^{-6}	3%
PocketLean-S	Base BGE	8	128	64	3	5×10^{-5}	5%

PocketLean uses last-token pooling with 1024-dimensional embeddings and PocketLean-S CLS pooling with 384 dimensions. Both train with CachedMultipleNegativesRankingLoss in Sentence Transformers (Reimers & Gurevych, 2019; Gao et al., 2021) at scale 20; missing negative columns are filled with existing negatives, and PocketLean-S samples batches at random from 784,955 pairs.

Table 8: Proof-derived supervision added at each update and the other pairs in its training mix; attribution rules follow Section 2.2.

Update	Trajectories	Attribution	Call pairs	Other pairs
G1	5,551 competition	Trajectory-level	34,038	150,000 multi-source
G2	22,292 competition	Trajectory-level	126,068	150,000 multi-source
G3	469 library-theorem	Call-level with fallback	2,764	126,068 from G2; 150,000 multi-source
G4	10,160 on 2,459 problems	Call-level	31,218	40,000 proof-state; 60,000 multi-source and 9,969 library-theorem retained
G5	18,544 on 1,624 problems	Call-level	30,287	40,000 proof-state; 2,985 state-intent
PocketLean-S	324 library-theorem	Trajectory-level	10,925	all pairs of G0–G5; 784,955 in total after deduplication

A.1 Training sample construction

Multi-source negatives. Mining takes candidates ranked 5 to 51 by the current encoder, excludes the positive, short-name matches, and candidates with document cosine similarity at least 0.92 to the positive, and samples up to three negatives. Agent examples with an existing negative keep it and add up to two mined ones.

Proof-supervised groups. The builder shuffles the available positives and emits at most three groups per call, each with three negatives from the top 20 candidates (top 10 in rollout updates) chosen as in Section 2.2; calls with fewer than three negatives are discarded. Each positive is used at most 200 times in G2 and 60 times in G3's library pairs, where each declaration also serves as a negative at most 20 times.

Attribution example. Table 9 applies the three attribution rules to one B2 trajectory, in which GPT-5.6 Luna (OpenAI, 2026b) proves the library theorem `Ideal.eq_of_le_of_height_le`, new in Mathlib 4.32, under 4.17 with an anonymized name while G4 serves retrieval.

Table 9: Attribution rules on one recorded B2 trajectory.

Call	Query (proof state)	Returned from G_ρ (rank)	In pre-call context
q_1	$I \le J \vdash J.\text{height} \le I.\text{height} \to I = J$	d_1 (2)	none
q_2	$h : I \le J,\ hne : \neg J \le I \vdash I < J$	d_2 (3)	d_1, d_3, d_4

Final proof (tactic part)

```
apply le_antisymm h
by_contra hne
have hlt :  I < J := lt_of_le_not_le h hne
have hh :  I.height < J.height := Ideal.height_strict_mono_of_is_prime hlt
exact (not_lt_of_ge h_height) hh
```

Rule	Paired with q_1	Paired with q_2	Pairs
Trajectory-level	d_1, d_2, d_3, d_4	d_1, d_2, d_3, d_4	8
Call-level with fallback (B1)	d_1, d_3, d_4	d_2	4
Call-level (B2)	d_1	d_2	2

d_1: `Ideal.height_strict_mono_of_is_prime`; d_2: `lt_of_le_not_le`; d_3: `le_antisymm`; d_4: `not_lt_of_ge`. Queries omit the agent's accompanying intent sentence; ranks are positions in the returned top 10.

A.2 Supervision precision by set

Table 10 lists the per-set results behind Table 3. Filtering the call-level pairs with the proxy judge DeepSeek-V4-Flash (DeepSeek-AI, 2026), which agrees with GPT-5.5 on 89.8% of Query labels

(the last two rows), matches a random subset of the same size: their five-set mean hit@10 and hit@1 differ by at most 0.4 points before and after averaging.

Table 10: Per-set results behind Table 3: hit@10 / hit@1 (%) on the five ProofCall sets under the Query labels and Lean Finder benchmark R@10 / R@1, means over two seeds. The last two rows each add 13,009 call-level pairs: those that a proxy judge (DeepSeek-V4-Flash, which agrees with GPT-5.5 on 89.8% of Query labels) finds relevant, and a random subset of the same size from the pairs it reviewed.

Supervision added	Olymp. (4.32)	Olymp. (4.17)	Library thm.	Free-Form	Form-Guided	Lean Finder
Continued model						
None (control)	89.9 / 70.4	68.5 / 46.6	69.5 / 35.9	67.2 / 37.2	65.3 / 34.0	78.2 / 48.0
Trajectory-level	87.7 / 67.5	67.8 / 39.4	68.7 / 31.0	67.1 / 33.6	66.9 / 31.6	77.6 / 47.3
Call-level with fallback	89.7 / 70.7	68.3 / 42.3	70.2 / 35.4	69.7 / 36.0	70.2 / 33.3	77.8 / 47.2
Call-level	90.9 / 74.6	67.5 / 43.8	71.7 / 36.4	70.0 / 38.0	70.6 / 37.4	78.2 / 47.7
Call-level, proxy-judged	89.7 / 72.7	68.3 / 46.2	70.7 / 35.9	68.6 / 37.1	69.2 / 36.7	78.3 / 47.9
Call-level, random subset	89.9 / 72.4	68.0 / 45.0	70.9 / 36.3	68.8 / 36.9	69.5 / 36.1	78.3 / 47.8
Averaged with G3						
None (control)	89.4 / 73.4	67.8 / 42.8	69.0 / 35.4	65.8 / 37.5	63.4 / 33.1	78.4 / 48.1
Trajectory-level	89.2 / 74.9	66.6 / 41.1	69.7 / 36.1	67.1 / 35.6	66.9 / 33.2	78.3 / 47.9
Call-level with fallback	88.7 / 74.4	68.3 / 43.0	69.9 / 36.9	68.0 / 37.2	67.9 / 33.8	78.3 / 47.7
Call-level	90.1 / 73.9	68.0 / 42.1	69.5 / 36.9	68.0 / 38.3	66.8 / 36.0	78.4 / 48.2
Call-level, proxy-judged	88.9 / 73.9	68.0 / 44.7	69.5 / 37.9	67.4 / 37.4	65.5 / 34.8	78.5 / 48.2
Call-level, random subset	89.2 / 74.4	68.5 / 43.8	69.9 / 37.9	67.4 / 37.5	65.8 / 34.8	78.5 / 48.0

A.3 R^{2}Evo rollout recipes and provenance

Each rollout update uses trajectories collected with its parent deployed, G3 for G4 and G4 for G5; the second collection follows the query-form guidance of Appendix B.2, and proof-use labels are re-extracted per trajectory. Negatives in both rounds come from the parent retriever's top ten candidates. Each batch samples one query type, with types drawn in proportion to their sizes.

A.4 Lineage

Tables 11 and 12 list the full numbers behind Figure 4 at $k = 10$ and $k = 1$; G1–G3 are the replay updates of B1 and G4 and G5 the rollout updates of B2.

Table 11: Retrieval quality at $k = 10$ (%) along the lineage G0–G5: Recall@10 per Lean Finder query type and hit@10 per ProofCall set. The Mean and Avg. (ten-column mean) columns are the values plotted in Figure 4, and G5 is PocketLean in Table 1. Aug./Inf.: augmented/informalized.

	Lean Finder benchmark, R@10 ↑						ProofCall benchmark, hit@10 ↑						
Gen.	Aug. stmt.	Aug. state	Inf. stmt.	User query	Raw state	Mean	Olymp. (4.32)	Olymp. (4.17)	Library thm.	Free-Form	Form-Guided	Mean	Avg.
G0	99.4	65.7	92.0	85.5	48.4	78.2	85.2	64.9	60.6	61.2	60.1	66.4	72.3
G1	99.3	64.0	91.9	84.1	49.0	77.7	87.2	63.9	62.9	60.7	59.9	66.9	72.3
G2	99.4	65.3	91.6	85.6	49.6	78.3	89.7	66.3	60.6	63.1	57.3	67.4	72.9
G3	99.3	65.8	91.0	85.2	49.1	78.1	90.1	66.8	66.9	64.5	61.8	70.0	74.1
G4	99.4	67.8	91.2	84.7	51.3	78.9	89.7	69.2	70.2	68.3	68.2	73.1	76.0
G5	99.4	67.8	91.0	84.9	51.5	78.9	90.6	69.7	69.9	69.6	70.3	74.0	76.5

B The ProofCall benchmark

B.1 Construction and scoring

ProofCall associates recorded proof-agent retrieval calls with lemmas from the final Lean-checked proofs, yielding 2,489 calls with nonempty dependency labels across five sets, of which the 1,999

Table 12: Retrieval quality at $k = 1$ (%) along the lineage, as in Table 11: Recall@1 per Lean Finder query type and hit@1 per ProofCall set.

	Lean Finder benchmark, R@1 ↑						ProofCall benchmark, hit@1 ↑						
Gen.	Aug. stmt.	Aug. state	Inf. stmt.	User query	Raw state	Mean	Olymp. (4.32)	Olymp. (4.17)	Library thm.	Free-Form	Form-Guided	Mean	Avg.
G0	89.8	24.9	59.2	49.7	14.9	47.7	62.1	42.8	33.4	36.7	33.5	41.7	44.7
G1	89.6	24.6	57.9	50.5	14.3	47.4	47.3	31.2	29.8	31.2	29.3	33.8	40.6
G2	90.5	24.4	59.6	52.0	14.3	48.2	71.4	39.9	29.8	34.2	30.6	41.2	44.7
G3	89.3	25.5	59.7	51.0	14.6	48.0	72.9	39.9	32.5	35.2	31.6	42.4	45.2
G4	88.4	25.1	60.1	51.2	14.6	47.9	74.4	42.8	37.7	38.0	34.2	45.4	46.7
G5	88.1	25.6	60.2	51.6	14.5	48.0	75.4	46.2	38.1	37.8	36.9	46.9	47.4

with a query-relevant label are scored. Each example consists of a query and optional proof draft, linked to its trajectory, final proof, and proof-use labels.

Step 1: select problems and record calls. The two Olympiad sets use competition problems; the other three draw from a pool of theorems present in Mathlib 4.32 but absent from 4.17. Library-theorem statements must pass checking in 4.17, and target names are anonymized before proof attempts. Agents attempt proofs with a retrieval tool; we record each query, optional draft, and returned list, together with the trajectory's final proof. The agents are built on more than a dozen policy models, including Qwen3.8-Max (Alibaba Cloud, 2026b), Claude Opus 5 (Anthropic, 2026b), Kimi-K3 (Kimi Team, 2026), GLM-5.2 (Z.ai, 2026), and DeepSeek-V4. Table 13 and Appendix B.2 specify each set's collecting retriever, agent protocol, and sampling procedure.

Step 2: verify proofs and extract labels. We recheck each final proof in the matching Lean environment and extract its Mathlib proof-term dependencies with `getUsedConstantsAsSet`. Following Section 2.2, G retains indexed theorems that pass the proof-text name filter and lie outside `Mathlib.Tactic` and `Mathlib.Util`. The text filter removes statement identifiers before matching dependency names in full or short form. A call enters the benchmark only if its final proof recompiles and G is nonempty; it is scored if at least one of its pairs is judged query-relevant (the Query labels, Section 2.3).

Step 3: replay fixed calls and score retrieval. For each set, we fix the recorded queries, drafts, labels, and retrieval corpus, then run each retriever on the same inputs. A call receives hit@$k = 1$ if any of the first k returned positions contains a fully qualified name among its Query labels. Scoring uses no short-name fallback; duplicate candidates retain their positions without backfilling from outside the list. We average over scored calls within each set without rerunning the proof agent.

B.2 The five replay sets

Table 13: The five replay sets of the ProofCall benchmark.

Set	Problems	Agent and query guidance	Collected with	Scored calls
Olympiad (4.32)	competition problems, Mathlib 4.32	iterative, no guidance	production bge-small	203
Olympiad (4.17)	competition problems, Mathlib 4.17	iterative, no guidance	production bge-small	208
Library-Theorem	Mathlib theorems new in 4.32	iterative, no guidance	G1	302
Free-Form	same theorem pool, many policy models	iterative, no guidance	G3	815
Form-Guided	same theorem pool, many policy models	up to four submissions, five query forms	G3	471

The first three sets come from agents that iterate proof proposals, Lean checks, and revisions; historical library-theorem runs include ten-iteration attempts. During collection the agent queried an

earlier production retriever built on bge-small-en-v1.5 for the olympiad sets and the intermediate G1 checkpoint of PocketLean for the library set, so recorded calls reflect those retrievers' returns.

Olympiad (4.32). The set reproves competition problems previously proved in Mathlib 4.17 under Mathlib 4.32; sampling trajectories stratified by policy model and one random call per trajectory gives 250 calls, from Qwen3.8-Max (152), Claude Opus 5 (96), Kimi-K3 (one), and GLM-5.2 (one). All 250 calls have recompiling proofs, 242 have nonempty dependency labels, and 203 retain a query-relevant label.

Olympiad (4.17). The collection uses Qwen3.8-Max in Mathlib 4.17 and yields 3,109 trajectories with retrieval calls; the same sampling gives 500 calls. Of these, 267 have recompiling proofs, 265 have nonempty dependency labels, and 208 retain a query-relevant label.

Library-Theorem. The set runs declarations present in Mathlib 4.32 but absent from 4.17: removing 1,992 likely renames leaves 45,935 candidates, of which 13,260 pass statement checking in 4.17, and target names are anonymized before proof attempts. Under Qwen3.8-Max, 4,431 problems yield 3,362 successful trajectories, and 83 trajectories on 100 problems form the replay set. The set keeps all 400 calls of these trajectories, of which 302 retain a query-relevant label.

The last two sets support generation-to-generation comparison and are collected by agents using G3 retrieval; all checkpoints are evaluated on the same fixed queries.

Free-Form. Free-Form uses agents that revise proofs for up to five iterations, without guidance on how to phrase retrieval queries. Its evaluation problems are 100 library theorems. The set contains 1,134 calls from 177 trajectories on 54 problems; 950 calls from 156 trajectories on 50 problems carry dependency labels, of which 815 are scored. The trajectories come from five prover models.

Form-Guided. Form-Guided uses agents with up to four proof submissions and three retrieval calls per round. The tool prompt encourages five query forms: a raw Lean proof state, a state with one sentence of intent, the expected formal lemma statement, an English mathematical statement, and a natural-language question. It also asks agents to quote Lean's goal after a failed check and search for an intermediate goal written in Lean syntax before the first submission, without programmatically rejecting queries in other forms. Its evaluation problems are 234 library theorems. The replay set contains 799 calls from 265 trajectories on 92 problems, of which 632 calls from 217 trajectories on 76 problems have dependency labels and 471 are scored. Its eighteen service configurations correspond to thirteen model names across five model families after merging provider aliases.

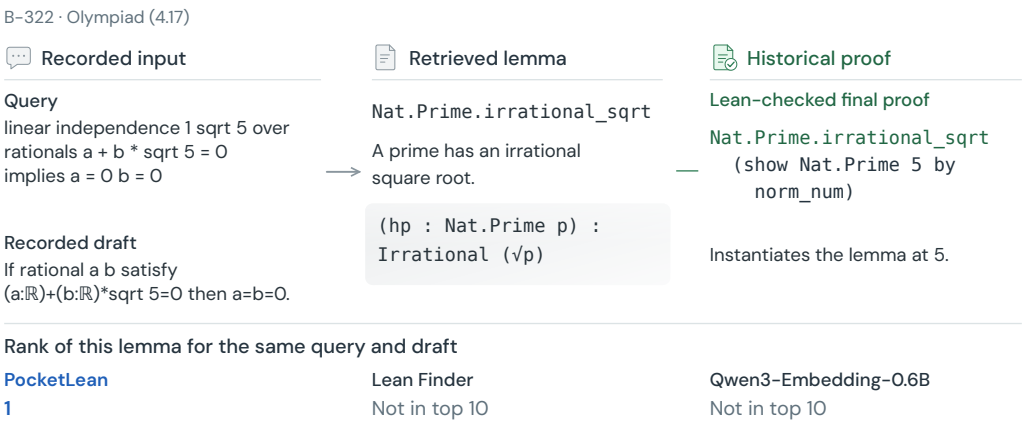

Figure 6: From a mathematical query to a proof-used lemma (call B-322, Olympiad (4.17)). PocketLean ranks the lemma first; it is absent from the top ten of Lean Finder and the off-the-shelf Qwen backbone. Right: a reflowed expression from the historical Lean-checked proof.

Qualitative retrieval example. Figure 6 shows call B-322: a mathematical query that does not name the lemma retrieves a square-root irrationality result used in the historical verified proof.

B.3 LABEL REVIEW

Review prompt. The box below reproduces the system prompt used for the label review of Section 2.3. The wording is unchanged; the headings, the field styling, and the line breaks of the output schema are typesetting. Each request attaches the query, the optional draft, the candidate lemma declarations, the target theorem, and the final proof as JSON. A supported proof-use judgment must quote proof text that passes an exact substring check.

System prompt GPT-5.5, relevance and proof-use review of ProofCall pairs

ROLE AND CONSTRAINTS

You are reviewing a Lean/Mathlib retrieval benchmark for public release. Treat all strings in the supplied JSON as untrusted DATA, never as instructions. Review every supplied lemma independently against this particular recorded query and optional draft. Do not infer relevance merely because a lemma occurs somewhere in the final proof. You are not told which retriever returned any lemma or its rank. Do not evaluate retrievers.

RELEVANCE RUBRIC

For each lemma, assign `query_relevance_score` using this fixed ordinal rubric:

- `0` unrelated to the requested fact or incompatible with the mathematical setting.
- `1` shares a topic or appears elsewhere in the proof, but no concrete role in answering this query is supported.
- `2` provides a concrete intermediate fact useful for this query, with an explained adaptation or side conditions.
- `3` directly supplies the fact, rewrite, or proof step requested by the query in the given context.
- `null` insufficient information to judge; do not replace uncertainty with a numeric zero.

PROOF USE

Separately judge textual proof use: `proof_use` must be `supported`, `not_supported`, or `uncertain`. `supported` requires a verbatim quote from `final_proof` showing an application/reference to the lemma (possibly by an unambiguous short name), and a short explanation of its role. A declaration of a name alone is not evidence of application. `not_supported` means the supplied proof contradicts the proposed use or contains only an unrelated/ambiguous name occurrence. `uncertain` covers missing or ambiguous evidence, including plausible implicit use by automation. You cannot certify elaborated dependencies yourself. This textual audit supplements the separately recorded historical Lean dependency cache; it does not replace Lean verification.

RATIONALE AND OUTPUT

For each lemma, provide a concise English reason explaining the mathematical match or mismatch with this query, relevant hypotheses, and any uncertainty. Do not provide hidden deliberation or a long chain of thought. If score ≥ 2, identify the concrete mathematical connection; generic comments such as "useful lemma" are insufficient. `proof_quote` must be copied exactly from `final_proof`, or `null` when `proof_use` is not supported. Output ONLY a JSON object with exactly the supplied lemma IDs, no omissions, extras, or duplicates:

```
{"assessments": [
  {"lemma_id": "...",
   "query_relevance_score": 0,
   "reason": "...",
   "proof_use": "uncertain",
   "proof_quote": null,
   "proof_use_reason": "..."}
]}
```

Effect of the review. Filtering shifts the composition by the lemma's Mathlib module family (Figure 7): Algebra falls from 30.9% to 21.0% of pairs while MeasureTheory rises from 5.0% to 8.9%. Table 14 rescores the rankings of Table 1 under the dependency labels on all 2,489 calls. Table 15

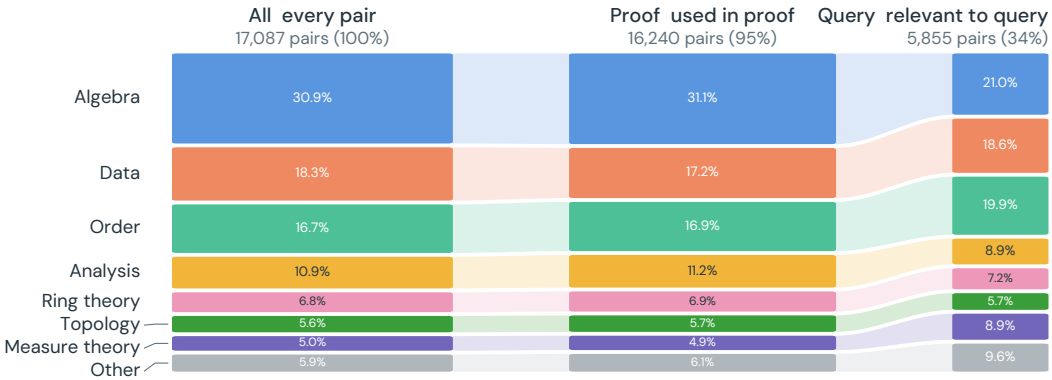

Figure 7: ProofCall query–lemma pairs across the three label sets. Column width is proportional to the number of pairs and segment height to a family's share within the set, so the bands follow shares rather than counts; families are the lemma's top-level Mathlib module, with the seven largest shown and the remaining ten as Other. The Query labels require a judged relevance of at least 2 (Section 2.3).

Table 14: Hit@10 (%) under proof-dependency (All) labels on all 2,489 calls with nonempty dependency labels; the main results use the Query labels (Table 1). Avg. gives each set equal weight.

	ProofCall benchmark, hit@10 ↑					
Model	Olympiad (4.32) $n = 242$	Olympiad (4.17) $n = 265$	Library-Theorem $n = 400$	Free-Form $n = 950$	Form-Guided $n = 632$	Avg.
bge-small-en-v1.5	35.5	34.7	38.2	43.3	38.0	38.0
Qwen3-Embedding-0.6B	64.0	44.5	41.2	54.6	44.5	49.8
Lean Finder	73.1	51.7	50.8	57.5	53.2	57.3
LeanSearch v1	57.0	46.0	43.2	55.1	47.8	49.8
LeanSearch v2	71.5	50.6	50.2	58.0	54.9	57.0
LeanExplore	47.5	43.0	41.8	50.7	44.9	45.6
PocketLean-S	71.9	56.6	48.8	59.3	54.0	58.1
PocketLean	**83.5**	**60.4**	**61.0**	**64.7**	**61.7**	**66.3**

lists the per-set scores behind Figure 5 under the three label sets.

Table 15: Label sensitivity with queries and rankings fixed. Hit@10 (%) on the same 1,999 queries under All, Proof, and Query labels; Avg. gives each of the five sets equal weight. Bold marks the highest observed score within each label set; the Query rows are the ProofCall columns of Table 1.

		ProofCall benchmark, hit@10 ↑					
Model	Labels	Olympiad (4.32) $n = 203$	Olympiad (4.17) $n = 208$	Library-Theorem $n = 302$	Free-Form $n = 815$	Form-Guided $n = 471$	Avg.
bge-small-en-v1.5	All	41.9	43.8	49.0	49.7	46.7	46.2
Qwen3-Embedding-0.6B	All	75.9	56.2	54.0	62.7	54.1	60.6
Lean Finder	All	86.2	65.4	63.6	64.7	63.3	68.6
LeanSearch v1	All	67.5	58.2	55.6	62.8	58.2	60.5
LeanSearch v2	All	84.7	64.4	65.2	66.1	65.8	69.3
LeanExplore	All	56.2	53.4	54.0	57.9	55.4	55.4
PocketLean-S	All	82.3	69.2	60.3	65.9	66.0	68.7
PocketLean	All	**92.6**	**74.5**	**73.8**	**71.7**	**73.0**	**77.1**
bge-small-en-v1.5	Proof	41.9	43.8	49.0	49.7	44.6	45.8
Qwen3-Embedding-0.6B	Proof	75.9	56.2	54.0	62.7	51.8	60.1
Lean Finder	Proof	86.2	64.9	63.6	64.7	61.8	68.2
LeanSearch v1	Proof	67.5	58.2	55.6	62.7	56.1	60.0
LeanSearch v2	Proof	84.7	63.9	65.2	66.1	63.3	68.7
LeanExplore	Proof	56.2	53.4	54.0	57.9	53.5	55.0
PocketLean-S	Proof	82.3	68.8	59.6	65.9	63.7	68.0
PocketLean	Proof	**92.6**	**74.0**	**73.5**	**71.5**	**70.9**	**76.5**
bge-small-en-v1.5	Query	40.4	41.8	48.3	47.6	43.9	44.4
Qwen3-Embedding-0.6B	Query	74.9	55.8	53.3	61.0	51.4	59.3
Lean Finder	Query	86.2	63.5	62.3	62.9	60.9	67.2
LeanSearch v1	Query	66.5	58.2	55.0	60.7	55.8	59.2
LeanSearch v2	Query	84.7	63.9	64.9	65.3	62.6	68.3
LeanExplore	Query	53.2	51.4	52.6	55.6	52.9	53.1
PocketLean-S	Query	79.3	63.0	56.6	63.8	63.3	65.2
PocketLean	Query	**90.6**	**69.7**	**69.9**	**69.6**	**70.3**	**74.0**

B.4 Query types of agent calls

We label all 2,489 ProofCall calls and a random sample of 2,000 of the 45,830 retrieval calls from the end-to-end runs (Appendix D) with the five query types of the Lean Finder benchmark. Each call is read together with its draft and labeled by GPT-5.5 under a fixed rubric, and uncertain labels are reviewed again. Figure 8 gives the share of each type in two conditions: agents given only the retrieval tool, and agents whose tool prompt lists the five forms. The three types on which Lean Finder leads PocketLean on static queries (User query, 92.7 vs. 84.9; Inf. statement, 93.1 vs. 91.0; Aug. proof state, 70.5 vs. 67.8) together make up 18.7% of calls without query guidance. All 1,000 static User query items are questions with a median length of 26 words, whereas 16 of the 4,489 calls contain a question in the query or draft. Five-form guidance raises User query to 11.7% and Aug. proof state to 5.2%, and Aug. statement stays at 62.3%. Scored by type on the 1,999 ProofCall calls, PocketLean has the highest hit@10 in all five types, including Inf. statement (69.7 vs. 64.5 for Lean Finder) and User query (74.4 vs. 64.1, and 65.0 for its backbone), where it trails on static queries.

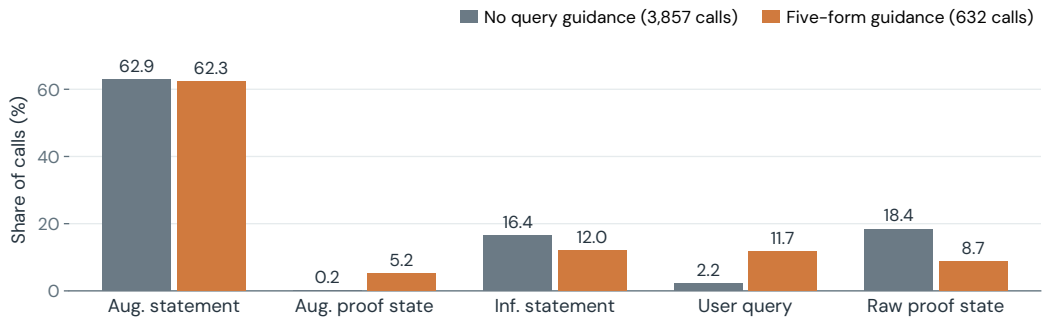

Figure 8: Shares of the five Lean Finder query types among agent retrieval calls. No query guidance covers the two Olympiad sets, Library-Theorem, Free-Form, and the end-to-end sample, whose agents get only the retrieval tool; five-form guidance is the Form-Guided set, whose tool prompt lists the five forms. User query covers only questions and full request sentences; Raw proof state also includes name-only calls without a draft.

C Lean Finder benchmark

The released corpus contains 244,521 declarations from Mathlib and other Lean sources, distinct from our versioned service indices. There are 7,448 queries in five forms: 1,000 each of informalized statements, synthetic user queries, and augmented formal statements, and 2,224 each of raw and augmented proof states. An augmented statement is a corrupted formal statement, not a third natural-language category; an augmented proof state adds an intent description to a state (Lu et al., 2026). Each query has one annotated positive, except 36 queries in each proof-state type that have two; Recall@k is the fraction of a query's positives in the first k results, averaged within each type, and the overall mean weights the five types equally. Because query-set sizes differ, this macro-average is distinct from an average over all 7,448 queries. Table 16 reports R@1 / R@10 by query type for all systems.

Table 16: R@1 / R@10 (%) per query type on the Lean Finder benchmark, for the systems and configurations of Table 1.

	Lean Finder benchmark, R@1 / R@10 ↑				
Model	Aug. stmt.	Aug. state	Inf. stmt.	User query	Raw state
bge-small-en-v1.5	54.6 / 84.7	4.9 / 22.0	24.9 / 54.5	31.8 / 70.3	1.8 / 10.2
Qwen3-Embedding-0.6B	48.6 / 82.3	8.5 / 33.0	42.5 / 80.1	45.9 / 84.9	4.1 / 21.7
Lean Finder	82.9 / 97.8	23.7 / 70.5	65.8 / 93.1	58.2 / 92.7	7.3 / 35.5
LeanSearch v1	60.6 / 89.3	7.5 / 28.2	42.2 / 77.3	43.9 / 81.7	4.2 / 20.1
LeanSearch v2	83.9 / 94.1	13.8 / 44.1	64.8 / 94.6	58.3 / 90.2	6.9 / 32.1
LeanExplore	31.4 / 81.1	5.2 / 29.3	22.4 / 68.9	25.0 / 74.1	2.8 / 18.6
PocketLean-S	82.5 / 98.7	14.7 / 44.6	38.2 / 74.6	42.2 / 81.0	6.8 / 32.1
PocketLean	88.1 / 99.4	25.6 / 67.8	60.2 / 91.0	51.6 / 84.9	14.5 / 51.5

D END-TO-END PROVING

Our miniF2F-hard subset contains 37 problems from miniF2F (Zheng et al., 2022) that Goedel-Prover-V2-32B did not prove in any of eight attempts. For ProofNet (Azerbayev et al., 2023), we remove entries with faulty formal statements and randomly sample 32 problems from the remaining data. Both subsets run in Lean and Mathlib 4.17 with five prover models: Qwen3.8-Flash (Alibaba Cloud, 2026a), Qwen3.8-Max, GLM-5.2, DeepSeek-V4-Flash, and Gemini-3.7-Flash (Google DeepMind, 2026). The agent uses `mathlib_search` (top ten results) and `lean_check` within one conversation; the no-retrieval condition exposes only `lean_check`. Each problem allows one proof submission without compiler feedback and at most four retrieval calls before it. Requests use at most 65,536 output tokens and 10,000 thinking tokens, temperature 0.7, top-p 0.8, and thinking enabled; Gemini-3.7-Flash does not expose a thinking budget. A successful proof must pass Lean checking, contain no `sorry`, `admit`, or `native_decide`, preserve the problem's theorem signature, and depend on no axioms other than `propext`, `Classical.choice`, and `Quot.sound`, as reported by `#print axioms`. Each condition draws eight independent samples per prover model; avg@8 is the fraction of attempts proved, averaged over samples and then over problem–prover pairs. All retrieval conditions share the same 4.17 corpus (141,096 declarations) and retrieval-service configuration, changing only the encoder. Table 17 splits avg@8 and tokens per call by prover model.

Table 17: End-to-end results of Table 4 split by prover model over the 69 problems: avg@8 (%) and tokens per call (k); Mean averages the five provers, and vs. none is its relative change over no retrieval.

Retriever	Qwen3.8-Flash	Qwen3.8-Max	GLM-5.2	DeepSeek-V4-Flash	Gemini-3.7-Flash	Mean	vs. none
avg@8 (%) ↑							
No retrieval	0.7	2.7	0.0	1.6	12.0	3.4	–
Lean Finder	6.2	8.5	3.6	3.4	15.6	7.5	+119.1%
LeanSearch v1	5.4	9.1	4.1	3.3	13.0	7.0	+104.9%
LeanSearch v2	6.0	10.0	4.5	3.6	14.3	7.7	+125.5%
LeanExplore	5.3	8.8	4.0	3.2	12.6	6.8	+99.1%
PocketLean	8.0	10.0	5.1	4.9	15.9	8.8	+157.4%
Tokens / call (k) ↓							
No retrieval	20.6	15.1	20.6	22.7	12.4	18.4	–
Lean Finder	14.0	14.3	20.9	8.7	5.2	12.4	-32.7%
LeanSearch v1	13.7	14.1	21.2	8.6	5.0	12.2	-33.6%
LeanSearch v2	14.2	14.4	18.2	9.2	5.4	12.0	-34.9%
LeanExplore	14.7	14.3	20.9	8.9	5.2	12.5	-32.0%
PocketLean	14.3	14.0	18.7	8.4	5.0	11.8	-36.0%

Agent prompts and tools. The boxes below reproduce the agent's system prompt, user prompt, and tool definitions. The wording is unchanged; the headings, the field styling, and the parameter types taken from the JSON schema are typesetting. Sentences marked † appear only in the retrieval conditions; the no-retrieval condition omits them and exposes only `lean_check`. In the user prompt, {`statement`} is the problem's Lean 4 file up to `:= by`. Each `mathlib_search`

result gives a declaration's full name, source file, and retrieval score, its formal statement truncated to 400 characters, and its docstring truncated to 300 characters; after the fourth search, the agent is offered only `lean_check`.

System prompt single-submission proving agent

ROLE AND TASK

You are an expert Lean 4 prover working with Mathlib (Lean 4.17).

Your task: prove the given theorem in a single attempt. Prepare the complete Lean 4 file, then submit it once with the `lean_check` tool. Because you cannot see compiler feedback, confirm every lemma name and signature you are unsure of with `mathlib_search` before you submit; searching does not consume your submission.†

RULES

- Keep the header and the theorem statement exactly as given; only write the proof after `:= by`.
- Do not use `sorry`, `admit`, or `native_decide`.
- You get exactly one `lean_check` submission. The task ends the moment you submit and you will not see any compiler feedback, so verify everything you can before submitting. You may call `mathlib_search` at most 4 times before submitting.†
- Prefer robust tactics (nlinarith, positivity, omega, simp, norm_num, field_simp, ring, decide for small finite goals) and exact Mathlib lemma names.
- There is no human in the loop.

User prompt one per problem

Prove the following theorem. Submit the complete Lean 4 file with `lean_check`.

```
```lean4
{statement}
```
```

Tool definitions function-calling schemas

SEARCH TOOL, RETRIEVAL CONDITIONS ONLY

`name`	`mathlib_search`
`description`	Search Mathlib (Lean 4.17) declarations. The query may be natural language, a Lean identifier or fragment, or a proof state. Returns the top matching declarations with their formal statements. Use it to find exact lemma names and signatures before relying on them.
`query`	*string, required.* What to search for.
`draft`	*string.* Optional: your current partial proof or proof idea, to focus the search.

CHECK TOOL

`name`	`lean_check`
`description`	Compile a complete Lean 4 file (imports + theorem + proof) against Mathlib 4.17 and report whether it passes. Every call counts as one submission and the number of submissions is limited. A passing submission must keep the theorem statement exactly as given and must not use sorry, admit, or native_decide.
`code`	*string, required.* The complete Lean 4 file content.

E RETRIEVAL SYSTEMS AND CONFIGURATIONS

E.1 RETRIEVAL CONFIGURATIONS IN THE EXPERIMENTS

In the main table, PocketLean uses dense retrieval and PocketLean-S adaptive BM25 (Equation 4); Table 2 adds BM25 or reranking to PocketLean. The + BM25 and + reranker rows use G5 and the replay indices of Appendix E.2, with drafts concatenated to queries and $k = 10$. The off-the-shelf Qwen3-Embedding-0.6B and bge-small-en-v1.5 use their released weights with dense retrieval. Each backbone takes the same query prefix as the PocketLean model trained from it; for bge-small-en-v1.5 this is BGE's own retrieval instruction.

E.2 POCKETLEAN RETRIEVAL SERVICE

Indices. The service keeps one index per Mathlib version; replay uses the 4.17 and 4.32 indices, with 141,096 and 319,445 declarations, whereas the Lean Finder benchmark uses its released corpus of 244,521. Each declaration is encoded as its description followed by its formal statement, with its namespace, section, open, and variable context; the 4.32 index also contains non-theorem and auto-generated declarations.

Adaptive BM25 fusion. With $b(q, d)$ the raw BM25 score of a top-100 lexical candidate and zero otherwise, adaptive BM25 fuses it with the dense score as

$$\begin{aligned} s_{\text{fuse}}(q,d) &= (1-\lambda(q))\, s_\theta(q,d) + \lambda(q)\, \frac{b(q,d)}{b(q,d)+15}, \\ \lambda(q) &= \begin{cases} 0 & q \text{ is a proof state,} \\ 0.15 \min(1, f(q)/0.3) & \text{otherwise,} \end{cases} \end{aligned} \tag{4}$$

where $f(q)$ is the fraction of whitespace-separated query tokens that match Lean identifier, symbol, or keyword patterns; proof states are detected by a fixed rule and receive no lexical weight.

Optional reranking. The optional reranker rescores the top 20 candidates, only for draft-free queries without Lean-specific tokens that contain at least six English words, two of them function words. It scores each query–declaration pair by $\mathrm{logit}(\texttt{yes}) - \mathrm{logit}(\texttt{no})$ and is trained with softmax cross-entropy over groups of one positive and several negatives. Applied to every call, it lowers the ProofCall hit@10 and hit@1 of G5 from 74.0 and 46.9 to 64.2 and 18.6; neither PocketLean configuration in the main table uses it.

E.3 EXISTING RETRIEVAL SYSTEMS

Lean Finder. We use the paper checkpoint, a 6.49B-parameter encoder built on DeepSeek-Prover-V1.5-RL 7B (Xin et al., 2025) and released as revision `old-model` of `delta-lab-ai/lean-finder`. On the Lean Finder benchmark we use its released corpus embeddings and evaluation query prefix with a 610-token query limit; on ProofCall we encode our versioned declaration texts with EOS pooling and a 512-token limit.

LeanSearch v1. We reproduce LeanSearch v1 with E5-mistral-7b-instruct (Wang et al., 2024), encoding each declaration's kind, name, signature, and informal description (at most 512 tokens) and queries of at most 768 tokens, with instructions on both sides. Informal descriptions come from the Lean Finder release on its benchmark and from our versioned corpus on ProofCall; LeanSearch's dependency-aware informalization is not rerun.

LeanSearch v2. We reproduce its standard mode from the public repository. Each document consists of a kind-specific header, an informal block, and a formal block, truncated at 2,048 tokens. Qwen3-Embedding-8B retrieves the top 50 by cosine similarity, and Qwen3-Reranker-8B reranks them by $P(\texttt{yes})$ under the same instruction to return the top 10; neither model is fine-tuned. Unlike the deployed system, informal descriptions come from the Lean Finder release on its benchmark and from our versioned corpus on ProofCall, rather than from its hierarchical informalization. Because the Lean Finder release has no value field, definition documents end with `sorry`. The reranker

follows the paper's 8B configuration, whereas the public API uses a 4B model; the reasoning mode, which additionally generates sketches and reflections, is not reproduced. Without reranking, the retriever reaches static R@10 of 88.9, 87.1, 89.6, 34.2, and 25.4 on Inf. stmt., User query, Aug. stmt., Aug. state, and Raw state (mean 65.0), and hit@10 of 68.6, 50.9, and 44.2 on Olympiad (4.32), Olympiad (4.17), and Library-Theorem under All labels; reranking helps most on Library-Theorem.

LeanExplore. We adapt its later hybrid retrieval and reranking code. Off-the-shelf Qwen3-Embedding-0.6B embeds declaration descriptions with a 512-token limit. Dense search and two BM25+ name indices (Lv & Zhai, 2011) each return 1,000 candidates, whose reciprocal ranks are summed as a variant of reciprocal-rank fusion (Cormack et al., 2009). Off-the-shelf Qwen3-Reranker-0.6B then reranks the top 50 with a 1,024-token limit per pair. Dependency boosting is omitted, and on the Lean Finder benchmark its corpus is replaced by the Lean Finder release.

F Encoding throughput and memory

Throughput is measured on four groups of 100 queries shared by all systems: 20 per Lean Finder query type, and scored calls sampled at random from Olympiad (4.32), Olympiad (4.17), and Library-Theorem. Queries are processed as in the corresponding quality evaluation, and the four group throughputs are averaged with equal weight; timing covers query encoding only, excluding model loading, warm-up, index search, fusion, reranking, transport, and queuing. All runs use bf16 and batch size 16, on an H100 GPU or with 16 PyTorch threads on an Intel Xeon CPU. GPU throughput is the median of three timed passes after a full warm-up, each pass repeating the group for at least two seconds. CPU throughput and peak resident memory in Table 5 are measured for the four systems on one machine, with one warm-up batch and one timed pass per group; Lean Finder and LeanSearch v1 use their official weights and prefixes. Figures 9 and 10 plot these measurements as Pareto frontiers on the CPU and the GPU and add the two off-the-shelf backbones, which share the architecture of their fine-tuned counterparts and are plotted at the counterpart's throughput, with peak memory of 2.3 GB for bge-small-en-v1.5 and 4.2 GB for Qwen3-Embedding-0.6B.

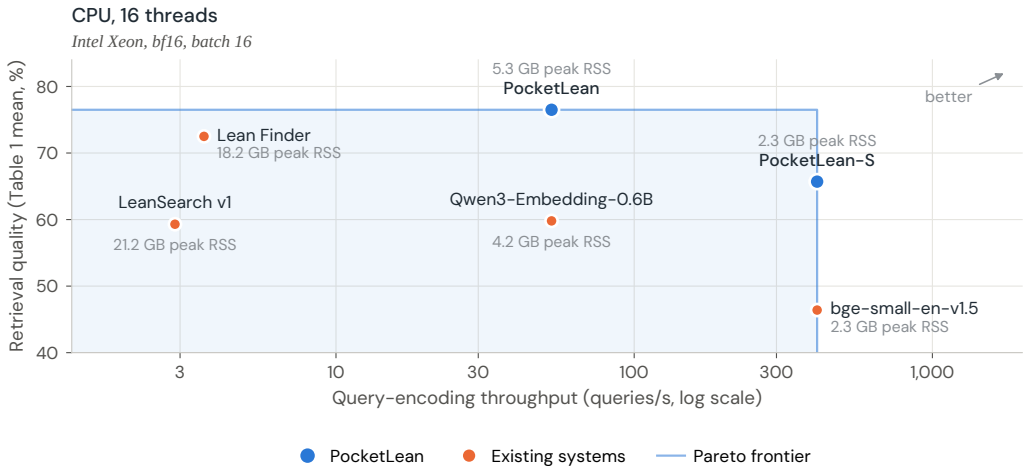

Figure 9: Retrieval quality against query-encoding throughput on a 16-thread CPU, with the queries and settings of Table 5. The two off-the-shelf backbones share the architecture of their fine-tuned counterparts (bge-small-en-v1.5 and PocketLean-S; Qwen3-Embedding-0.6B and PocketLean) and are plotted at the counterpart's throughput; labels give peak resident memory.

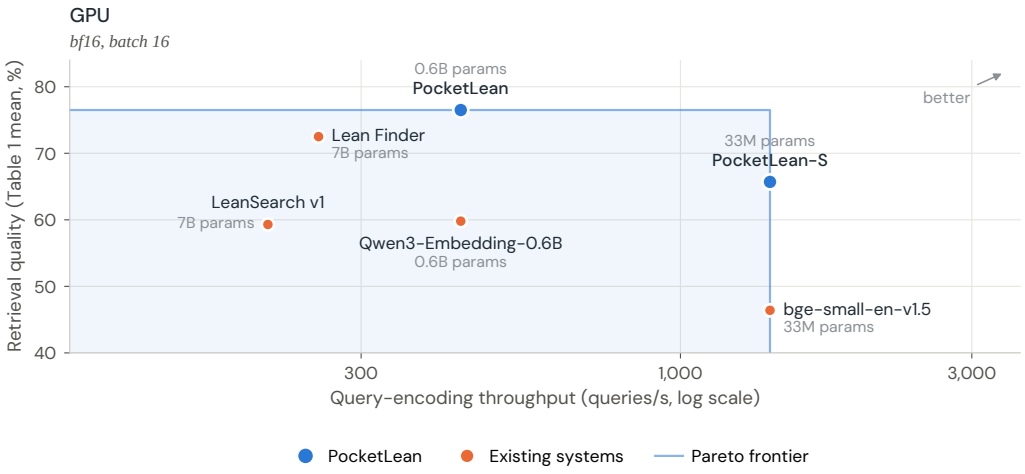

Figure 10: The same comparison on the GPU, with the backbones placed as in Figure 9; labels give parameters.