THEOREMGRAPH: BRIDGING FORMAL AND INFORMAL MATHEMATICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mathematical knowledge is organized around statements and their dependencies, but this structure is exposed unevenly: informal papers cite mostly at the document level, while formal libraries record fine-grained dependencies over a much smaller body of mathematics. We introduce **TheoremGraph**, a unified statement-level dependency graph spanning both: 11.7M theorem-like environments parsed from mathematics arXiv with 18.3M candidate directed dependencies, each labeled by the extractor that proposed it, and **LeanGraph**, a Lean 4 elaborator-level extractor producing 388,105 declaration nodes and 11.3M typed edges across 25 Lean projects. We bridge the two graphs by embedding generated natural-language slogans into a shared semantic space; an LLM judge affirms 47,952 (informal, formal) matches among the candidates above a 0.8 cosine-similarity floor (48% of those judged), with the acceptance rate rising to 87% for scores ≥ 0.9. On the MathlibQR fair-810 retrieval benchmark, our name-and-signature retriever, reranked with the same Qwen3-Reranker-8B protocol as LeanSearch v2, reaches 0.883 Recall@10 / 0.739 nDCG@10, +10.3pp / +11.6pp over its reranked system, a comparison we verify by reproducing its reported numbers with our code. A recall-tuned variant of our retriever comes within 0.5pp of that reranked recall with no reranker at all. On 24 held-out Mathlib theorems, retrieval raises the number of correctly autoformalized statements from 5 to 8, using a quarter of the tokens of library search. We release the dataset, extractors, HTTP API, and MCP interface as infrastructure for mathematical search, attribution, and agentic formalization.

1 INTRODUCTION

Mathematical knowledge is built from discrete statements and their dependencies, but contemporary practice hides this structure: authors cite papers rather than the exact lemmas, definitions, or theorems a proof requires, which coarsens attribution and lets duplicated work go unnoticed. In a large study of withdrawn arXiv submissions, 2.4% of categorized withdrawals are self-identified as "not novel" (Rao et al., 2024). One solution to the attribution problem is a proof assistant. In Lean (de Moura & Ullrich, 2021), an interactive theorem prover, a proof must specify every logical dependency precisely enough to be type-checked by a trusted kernel. But formalized mathematics remains far smaller than the informal literature. Lean's largest mathematics library, Mathlib4 (The mathlib Community, 2020), holds over 400K declarations (The mathlib Community, 2026), concentrated in foundational and undergraduate-level material, limiting how far formal methods can navigate or verify contemporary mathematics.

This gap has started to close quickly, and a graph of statements is what made that possible. In September 2026, Anthropic reported the first end-to-end computer-checked proof of Fermat's Last Theorem in Lean, produced by dozens of Claude agents working collaboratively (Anthropic, 2026). In eleven days, the agents formalized 29,511 theorems in 13M lines of Lean. The Lean community had expected the project to take years, and its blueprint for the proof alone runs to 86 pages (Buzzard & contributors, 2026). To achieve this, the agents collaborated via Prove2Me (Chen et al., 2026), an open platform for multi-agent formalization.

What the platform added, which more compute alone had not, was structure. Anthropic's writeup credits three features as direct unlocks: (1) a directed acyclic graph of theorem statements that tells agents what to prove next; (2) statements and proofs kept in separate files, with the links between

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.

them tracked independently; and (3) a natural-language description on every statement, so that agents can search for and reuse existing lemmas. Prove2Me's graph covers one theorem. Its authors name the next bottleneck themselves: "how agents should search a large, evolving corpus of formal statements" (Chen et al., 2026). TheoremGraph is the same object built for the mathematical literature as a whole, with that search problem as its first-class concern (§5.2). The proof itself is public, so we can also apply the same tools to it: we rebuild its statement graph, look for redundancy inside it, and compare it with the existing Lean corpus (§4).

We build and release the bridge between these two worlds. **TheoremGraph** is a statement-level dependency graph spanning informal and formal mathematics: an informal graph of 11.7M arXiv statements with 18.3M extracted dependencies (§2.1), a formal graph of 388,105 Lean declarations with 11.3M typed edges extracted by our **LeanGraph** tool (§2.2), and a shared slogan-embedding space (§2.3) that lets us join the two graphs by search followed by an LLM judge (§3). We validate the graph on two downstream tasks at the core of AI-assisted formalization: retrieval-augmented autoformalization, i.e. specification generation from informal statements (§5.1), and formal concept retrieval against LeanSearch v2, where our approach beats LSv2's full reranked system under its own reranker (§5.2). Extended related work appears in Appendix A; in brief, we extend sloganized statement retrieval (Gao et al., 2024; 2026; Welleck et al., 2021; Alexander et al., 2026) with explicit dependency structure, and extend graph-based analyses of formal libraries (Li et al., 2026; Bauer et al., 2023; Mishra et al., 2024; Bian et al., 2025) across the informal/formal divide. The closest formal-side precedent is LeanDojo (Yang et al., 2023), whose authors extract premise dependencies from Mathlib and train a retrieval-augmented prover, ReProver, on them. In contrast, we train no prover: our formal graph adds typed edges across 25 projects and links them to informal statements. For autoformalization, we build on the evaluation settings of Wu et al. (2022), ProofNet (Azerbayev et al., 2023), and Herald (Gao et al., 2025), and use the graph as a retrieval source rather than as training data.

2 Graph construction

2.1 Informal graph

Our corpus is the Kaggle arXiv metadata snapshot filtered to mathematics-tagged papers, with references resolved through Semantic Scholar and LaTeX source retained.[1] Statement identification follows the regex strategy of Alexander et al. (2026): a parser over the LaTeX source identifies theorem-like environments and records each statement's type, reference number, `\label` key, body, proof, and local context, discarding malformed statements. This yields 11.7M statements. Three extractors then propose directed dependency edges. The *deterministic* extractor resolves `\ref`-like commands within a paper and `\cite` keys against its reference list, down to a specific statement when the citation names one. The *heuristic* extractor adds edges from nearby references, backward discourse cues, and prose mentions such as "Theorem 3.2." The *notation* extractor uses Qwen3-235B-A22B-Instruct-2507 to link each statement's notation to the closest prior compatible definition. Together they recover 18.3M edges (14.9M within-paper, 3.5M external), and every released edge is tagged with the extractors that proposed it.

We validate the pipeline on 500 sampled papers against an independent LLM judge (Kimi K2.5) (Table 1). The judge first lists the within-paper dependencies it finds on its own, then checks each extractor edge it did not list. Deterministic edges reach 98.8% judge-verified precision; per-edge extractor labels let users pick their own precision–coverage tradeoff.

2.2 Formal graph: LeanGraph

LeanGraph is a Lean 4 elaborator-level extractor that builds typed declaration-dependency graphs from compiled Lean projects, running inside Lean over the kernel `Environment` API. Extracting from elaborated, type-checked declarations rather than source text gives access to post-elaboration constants and proof terms that source-level parsers cannot reliably recover. Each dependency is assigned one of six semantic edge types (`extends`, `field`, `sig`, `proof`, `def`, and `docref`), separating structural, type-level, value-level, and documentation-level dependencies so downstream

[1] March 2026 snapshot; the newest paper in the corpus is arXiv:2603.05505.

Table 1: Extracted informal dependency edges vs. an independent LLM judge on 500 arXiv papers. Per-extractor rows may overlap; "Any" deduplicates. The judge proposes 6,372 edges no extractor recovered; treating it as ground truth gives $F_1 = 0.642$.

Extractor	Edges	Judge-Agreed	Judge-Rejected
Deterministic	5,051	4,989 (98.8%)	62 (1.2%)
Heuristic	4,616	3,535 (76.6%)	1,081 (23.4%)
Notation	6,241	2,665 (42.7%)	3,576 (57.3%)
Any	14,481	9,855 (68.1%)	4,626 (31.9%)

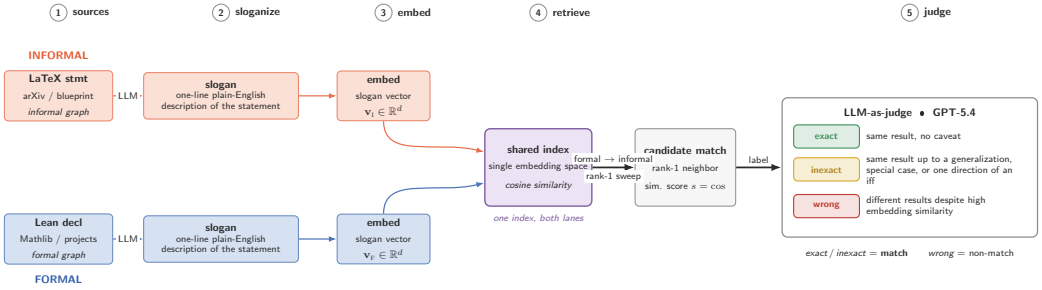


Figure 1: The matching pipeline. Lean declarations and LaTeX statements are each sloganized and embedded into one shared index; a formal→informal rank-1 cosine query proposes candidate matches, which a single GPT-5.4 judge labels `exact` / `inexact` / `wrong` (`exact`/`inexact` count as matches).

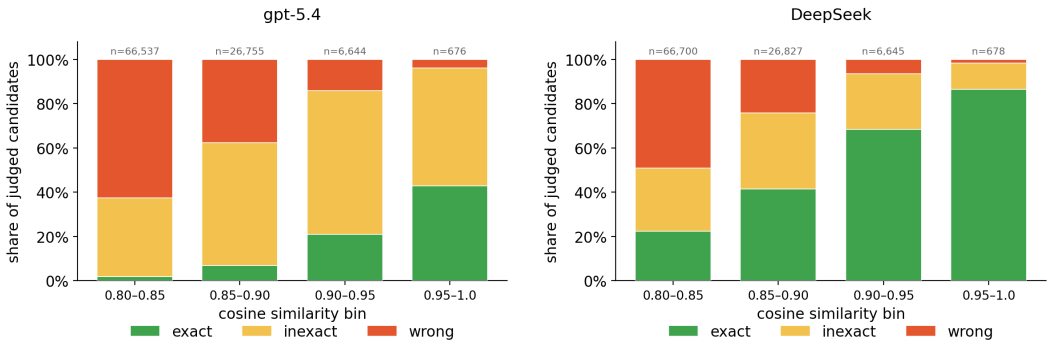
systems can rank or filter edges by role. A structural inclusion predicate based on Lean's classification APIs and doc-gen4's renderability criteria (Lean Prover Community, 2026) removes kernel-generated artifacts while retaining user-facing declarations. Across 25 Lean projects, LeanGraph extracts 388,105 nodes and 11.3M edges; the Mathlib v4.27–v4.29 graph alone contains 351K nodes after this filter and 9.3M within-library edges. Appendix A positions LeanGraph against prior Lean graph tools (frenzymath, 2024; Yanahama & Sannai, 2026).

2.3 A SHARED SEMANTIC SPACE

The two graphs differ sharply in density (edge-to-statement ratio $\sim$1.56 informal vs. $\sim$29.2 formal) and share no explicit edges. To link them, we map every statement into a shared semantic space. Following statement-level retrieval work (Gao et al., 2024; Alexander et al., 2026), each statement gets a *slogan*: a concise, standalone natural-language summary generated by Qwen3-235B-A22B-Instruct-2507 (Qwen Team, 2025). For informal statements, the prompt starts from the statement alone and adds context in stages (the proof and local context, then neighboring statements and dependencies) only when the model answers that its context is insufficient; a final stage forces a best-guess slogan, so every one of the 11.7M statements gets one. For formal declarations, the prompt has a fixed size, so we fill it with the most important dependencies first, ranked by the semantic edge roles of §2.2. Slogans are embedded into $\mathbb{R}^{4096}$ with Qwen3-Embedding-8B (Zhang et al., 2025), ℓ_2-normalized, and stored under an HNSW index (Malkov & Yashunin, 2018) with a binary-quantized projection for fast candidate generation.

3 BRIDGING THE CORPORA

We look for **matches** (Figure 1): (informal, formal) statement pairs expressing the same mathematical result under the same assumptions. Author-annotated Lean blueprints (Massot, 2020), the documents in which a formalization project pairs each Lean declaration with the informal statement it formalizes, supply 1,595 such pairs. Finding matches at scale would give autoformalization systems the same kind of scaffold. It would also help with the "dependency issue" in automated theorem proving,



(a) GPT-5.4, our judge of record.

(b) DeepSeek, a more lenient high-recall comparison.

Figure 2: Verdict composition by cosine-similarity bin, over every candidate at similarity ≥ 0.8. Both judges shift from `wrong` to `exact` and `inexact` as similarity rises; DeepSeek is consistently more generous, so we keep GPT-5.4 as the stricter set of record. Below the 0.8 floor, a pilot of 150 random candidates per bin found 14% matches at 0.7–0.8 and 3% at 0.6–0.7.

where progress "typically hinges on synthesizing insights across a multitude of related papers" (Chen et al., 2025).

Setup. We sweep 385,657 formal declarations: the 388,105 in the formal graph minus 2,448 constructors and compiler-generated names, which have no natural-language counterpart. For each, we look up the informal slogan with the highest cosine similarity in the shared index. This is a nearest-neighbor query across the two corpora, and the neighbor it returns, together with its similarity, is a **candidate match**. The index is approximate, and a parameter, `ann_k`, sets how many neighbors it examines for each query. At `ann_k` $= 50$, a candidate is found for 55.3% of declarations. Raising `ann_k` to 500–1000 for 400 of the unmatched declarations almost never surfaces a candidate above similarity 0.85, so a missing candidate almost always reflects the absence of a close informal counterpart. We validate the retrieval by simulating search over the blueprint pairs. When we query with each pair's formal slogan, the annotated informal partner ranks first 43.5% of the time, and it appears in the top ten 69.9% of the time. When the top result is instead a different candidate with similarity at least 0.85, an LLM judge rates 65% of those cases (n=187) as valid alternative restatements.

Judging. A GPT-5.4 judge sees both slogans, the declaration name and Lean signature, and the informal LaTeX, and labels each candidate as `exact` (citable as the same result), `inexact` (the same underlying result, such as a generalization, a special case, or one direction of an equivalence), or `wrong` (different results despite high similarity). We count `exact` and `inexact` as matches. GPT-5.4 is the judge of record because a domain expert found our initial choice, Opus 4.8, over-generous, and a curated set should favor precision. The judge is stable under re-runs, with κ=0.86 on a 500-candidate resample.

Where to judge. Before judging at scale, we measured how the match rate falls with similarity by judging 150 random candidates in each 0.1-wide similarity band. The fall is steep: 14% of candidates in 0.7–0.8 are matches, and 3% in 0.6–0.7, so judging below 0.8 would mostly confirm non-matches. We therefore judge every candidate at similarity 0.8 and above. Within that range the rate keeps climbing, from 37% in 0.80–0.85 to 96% above 0.95 (Figure 2), so similarity itself is a reliable confidence score for the released matches.

Results. The retrieval pool holds 100,865 candidates above the floor; 100,799 of them received a usable label (66 were not submitted or came back unparseable). Of these, GPT-5.4 affirms 47,952 as matches, a rate of 48%. Quality concentrates at the high-similarity end: the ≥ 0.9 tier alone contributes 6,353 matches, with an acceptance rate of 87%. A more lenient second judge, DeepSeek-V4-Pro, affirms 61,234 candidates (61%). We keep the stricter GPT-5.4 set as our released matches, and report the DeepSeek set as a high-recall comparison.

Table 2: Audit of the FLT proof. A *repeat* is a theorem whose statement matches another theorem in the same corpus except for its name. A *twin* is a theorem whose statement embedding has cosine similarity at least 0.99 with another theorem's. FLT (human) is the Imperial FLT project together with the separate flt-regular project (Best et al., 2025). Mathlib counts are a lower bound, since the proof builds on a newer Mathlib than our index.

Citation graph of the FLT proof			
Theorems / citation edges			29,511 / 106,897
Lemmas cited exactly once			15,651 (53%)
Lemmas cited 20 or more times			775
Redundancy	FLT (agents)	FLT (human)	Mathlib sample
Theorems that repeat another statement	0.45%	0.73%	0.47%
Theorems with a near-variant twin	7.7%	5.6%	2.1%
Matches against prior work			
Top hit from FLT (human), rank 1 / top 10 (n=50)			64% / 88%
Agents' FLT lemmas with a Mathlib hit above 0.85			60
judged exact / inexact restatements			0 / 23

4 AUDITING AN AGENT-BUILT PROOF

Anthropic's FLT proof, which its authors describe as the largest Lean proof ever constructed, is public. Each theorem sits in its own file, and each proof file imports exactly the theorems it uses, so the citation graph can be read directly from the files. We embed every statement, then search it against the rest of the proof and against TheoremGraph's formal corpus (Table 2). We ask two questions: did the agents prove the same result twice, and can we tell where a renamed theorem came from?

Repeated proofs. The agents mostly reused their own lemmas well: 775 lemmas are each cited 20 or more times. But 67 pairs of theorems state exactly the same thing under different names, and each was proved separately. For example, these two theorems both say that every weight-2 cusp form for $\Gamma_0(1)$ is zero, and neither cites the other:

```
CuspForm.eq_zero_of_gamma0_one_weight_two
    ModularForm.S2_Gamma0_one_eq_zero
```

The agents were told to search before proving something new, and these repeats slipped through because their names differ. This is not unusual for Lean. By a comparable measure, 0.73% of theorems in the human-written Imperial FLT and flt-regular projects and 0.47% of a sample of 29,511 Mathlib theorems also repeat another statement, against 0.45% in the agents' proof, and near-duplicates that differ by one hypothesis are common in all three (Table 2).[2] So the agents' redundancy is of the same order as human authors'. In every corpus the repeats carry different names, and only a search by statement finds them.

Attribution. Anthropic lists which parts of the proof were copied or adapted from earlier projects, found by comparing text line by line. Its authors note that this misses results restated in different words. Statement search does not depend on wording, so we tested whether it recovers known sources. We took 50 theorems that Anthropic's list traces to the human-written Imperial FLT project or to flt-regular (Best et al., 2025) and searched TheoremGraph with each theorem's English description. For 64% of theorems the top result comes from these projects, and for 88% one of the top ten does. We also looked for lemmas in the agents' proof that re-prove something already in Mathlib. Only 60 have a close Mathlib match (similarity above 0.85). GPT-5.4 judges 23 of these to be the same result with small differences, and none to be identical. This is close to the 22 Mathlib re-proofs Anthropic reports, and it is a lower bound, since our Mathlib index is older than the one the proof uses.

[2]Each corpus is compared with itself. Human-written FLT and Mathlib statements come from our index as pretty-printed signatures, which can hide type coercions, so some of their repeats (for example, the same lemma stated for two integer types) are not true duplicates. The agents' FLT statements are source text. The rates are therefore comparable only in order of magnitude.

Table 3: Statement-only autoformalization of 24 Mathlib v4.30 theorems, retrieving over v4.29 (`claude-opus-4-7`-judged, hand-checked).

Condition	Typecheck	Strict	Eval. correct	Tokens	Tool calls
None	22/24	4/24	5/24	52k	276
RAG	20/24	6/24	**8/24**	**14k**	**68**
Library	23/24	5/24	6/24	52k	275
RAG+Library	**24/24**	**7/24**	**8/24**	37k	188

5 Downstream evaluations

5.1 Retrieval-augmented autoformalization

We test whether slogan retrieval helps an LLM autoformalize natural-language statements into Lean. Given an informal description, `claude-sonnet-4-6` generates a single Lean 4 declaration ending in `:= sorry`, with at most three `typecheck` calls per target. To reduce memorization, we evaluate on 24 theorems introduced in Mathlib v4.30 while retrieving only from the v4.29 corpus; queries are `qwen3-8b` back-translations of the gold signature with the declaration name removed. The retriever uses slogan embeddings with a query head trained on typed dependency labels. We compare four conditions: *None* (informal statement only), *RAG* (top retrieved premises added), *Library* (a `grep` tool over the full v4.29 declaration listing), and *RAG+Library*. A hand-checked `claude-opus-4-7` judge labels each output *strict* (same proposition up to notation) or *evaluated correct* (high-confidence equivalent).

Table 3 reports, for each condition, how many of the 24 outputs typecheck, how many the judge marks strict, how many it marks evaluated correct, and the total tokens and tool calls the model spent. Retrieval improves evaluated correctness on the 24 held-out Mathlib v4.30 theorems from 5/24 to 8/24, and it does so with about a quarter of the tokens and tool calls of library search, because the retrieved premises replace most of the model's own exploration of the library. Typechecking alone is not a reliable success signal: the ungrounded condition typechecks 22/24 outputs but is evaluated correct on only 5/24. About 10/24 targets fail in every condition, suggesting some informal queries underdetermine the intended Lean statement.

5.2 Formal concept retrieval vs. LeanSearch v2

Benchmark. We compare with LeanSearch v2 (LSv2) (Gao et al., 2026) on MathlibQR fair-810: 810 informal queries for 171 Mathlib declarations that every compared system indexes. Both systems use the same embedder, Qwen3-Embedding-8B, without fine-tuning, so differences come from what each system embeds. LSv2 embeds a passage with the declaration's kind, type signature, and an informal description. Our baseline embeds only the slogan, and trails LSv2's retriever by 7.1pp Recall@10 and 11.4pp nDCG@10 (Table 4). The reason is that many declarations, such as `Lattice` or `Functor`, essentially *are* their names, and a one-line summary gives a query for a name nothing to match.

Closing the gap. We test four changes, combined in different ways across configurations A–F (Table 4):

1. a second embedded passage with the declaration's name, signature, and dependencies, plus a name index;
2. HyDE query rewriting (Gao et al., 2022), which also searches with a drafted hypothetical statement;
3. graph expansion, which adds each result's parents in the dependency graph, since name-like queries often want the parent (LSv2 builds this graph but does not use it for retrieval);
4. a wider shortlist.

The name-and-signature passage matters most. B and F use the same search settings and differ only in the passage, and F beats LSv2's retriever on every metric. Using both passages at once raises recall but hurts ranking, likely because a second passage per declaration lets near misses into the top ranks (compare D and F). With both passages and graph expansion, configuration E matches LSv2's

Table 4: MathlibQR fair-810 results. Each row is one configuration; checkmarks show which components it uses. *Slogan* and *Name/sig* are the two passages we can embed per declaration. *Search* is a name index, HyDE query rewriting, and a wider shortlist, used together. *Graph* adds each result's dependency-graph parents. † Reranked with Qwen3-Reranker-8B under LSv2's protocol. LSv2 rows are from Gao et al. (2026).

	Components				Metrics			
Configuration	Slogan	Name/sig	Search	Graph	nDCG@1	nDCG@5	nDCG@10	R@10
Ours (A) Baseline	✓				0.201	0.345	0.380	0.586
Ours (B)	✓		✓		0.262	0.441	0.467	0.680
Ours (C)	✓		✓	✓	0.319	0.461	0.504	0.716
Ours (D)	✓	✓	✓		0.343	0.516	0.550	0.767
Ours (E) *recall-opt.*	✓	✓	✓	✓	0.349	0.504	0.548	0.775
Ours (F) *ranking-opt.*		✓	✓		0.391	0.534	0.558	0.733
LSv2 retriever					0.340	0.472	0.494	0.657
LSv2 retriever + reranker†					0.470	0.601	0.623	0.780
Ours (F) + reranker†, top-50					0.578	0.701	0.715	0.843
Ours (F) + reranker†, top-200					**0.591**	**0.721**	**0.739**	**0.883**

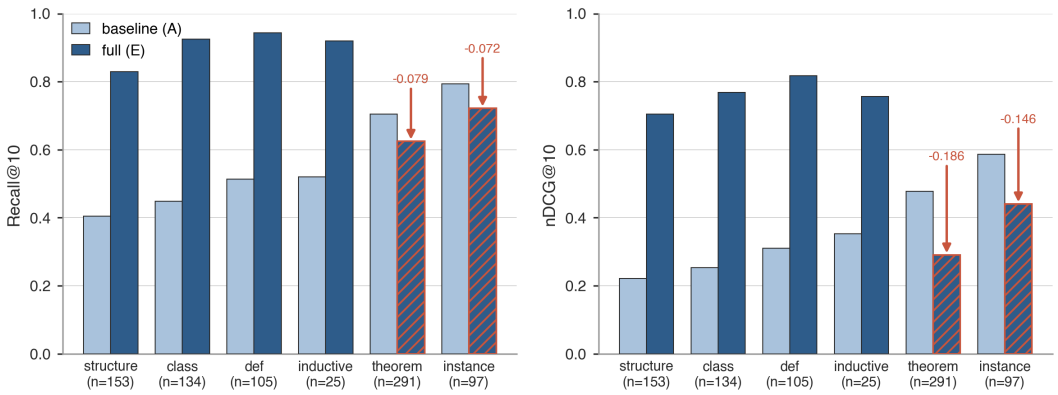


Figure 3: MathlibQR fair-810, Recall@10 by declaration kind, comparing the baseline (Config A) with the full lever stack (Config E). The interventions concentrate their gains on declarations identified mainly by name (structures, classes, definitions, and inductives), while `theorem` and `instance` were already well served by slogans.

reranked recall (0.775 vs. 0.780) without any reranker. The gains are largest on the declaration kinds that slogans handled worst (Figure 3).

Reranking. We then rerank F's top candidates with Qwen3-Reranker-8B, copying LSv2's protocol exactly: the same prompt, the same yes/no relevance score, and ranking by that score alone. With the same reranker, our system beats LSv2's reranked system by 10.3pp Recall@10 and 11.6pp nDCG@10 (0.883 / 0.739 against 0.780 / 0.623; Table 4). The gap holds at both reranking depths and lies outside our confidence intervals (Appendix D).

Is the comparison fair? LSv2's numbers come from its paper, so we checked that our code is not the source of the gap. We ran LSv2's full pipeline ourselves, from its released corpus and prompts, using our own code. It reproduces LSv2's reported reranked numbers to within 0.3pp (Appendix D). The two reranked systems therefore share the reranker, the code, and the depth, and the difference comes from the candidates each retriever finds.

Limits. These retrieval settings do not help chained-premise retrieval: on MathlibMPR, without reranking, they lower group Recall@10 from 0.224 to 0.165 against our untuned baseline. We have not run the reranked system on MathlibMPR.

6 CONCLUSION

TheoremGraph links 11.7M informal arXiv statements with the Lean ecosystem through a shared slogan-embedding space, yielding 47,952 LLM-affirmed cross-formality matches, a small autoformalization case study in which retrieval cuts cost by three quarters, and a MathlibQR fair-810 candidate pool that beats LeanSearch v2's reranked system by +10.3pp Recall@10 / +11.6pp nDCG@10 under the same reranker. The FLT formalization suggests that a statement graph with natural-language descriptions is what lets agent fleets formalize at scale (Anthropic, 2026), and our audit of it shows where search still falls short; TheoremGraph supplies that substrate for the literature rather than for one theorem. The dataset, extractors, HTTP API, and MCP interface are all released;[3] limitations are discussed in Appendix E. Two further experiments appear in the appendix: an in-context literature-search test on redacted arXiv passages (Appendix B) and the relation between graph distance and embedding distance (Appendix C).

ETHICS STATEMENT

This work builds retrieval infrastructure over publicly available mathematical text (arXiv LaTeX source and open-source Lean libraries). No human-subject data is involved. arXiv licensing is respected per record: the public release of judged matches is restricted to statements carrying a redistributable license, and every released record carries its source license (Appendix E).

REPRODUCIBILITY STATEMENT

Section 3 gives the retrieval, judging, and threshold protocol for the cross-formality matches, including the yield experiment that fixed the judging floor (Figure 2). Section 5.2 specifies the benchmark split (MathlibQR fair-810), the embedder, every retrieval intervention, and the exact reranker scoring recipe, with all retrieval and reranking traces, an independent scorer, the bootstrap script, and our end-to-end reproduction of LeanSearch v2 included in the anonymized supplementary material. Section 4 describes the FLT audit inputs, which are derived from the public proof repository. The supplementary material also contains the FLT audit scripts, the audited pairs with both judges' labels, and the twin-rate baselines. The dataset, both extractors, the HTTP API, and the MCP interface are released (links withheld for anonymous review; see Appendix E).

AI USE STATEMENT

In this work, we used generative AI tools as components of the method: LLM-generated slogans, LLM judges for informal dependency edges and for cross-formality matching, an LLM judge for the FLT audit samples, LLM-based query rewriting, and an LLM reranker; each is named with its role in Sections 2.1–5. We also used AI coding assistants to implement analysis scripts and the figure code, to help design the FLT audit experiments, and to draft and edit parts of the manuscript text. We have not used generative AI tools to generate synthetic datasets, formulate or prove mathematical claims, or interpret results without author review. We have reviewed all AI-assisted work: every reported number was recomputed from the raw retrieval and reranking traces by an independent script, LLM-judged samples are labeled as such, and all text was reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Akiko Aizawa, Michael Kohlhase, Iadh Ounis, and Moritz Schubotz. NTCIR-11 Math-2 task overview. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies*, 2014.

[3] Links withheld for anonymous review.

Jesse Alama, Lionel Mamane, and Josef Urban. Dependencies in formal mathematics: Applications and extraction for Coq and Mizar, 2012. URL https://arxiv.org/abs/1109.3687.

Luke Alexander, Eric Leonen, Sophie Szeto, Artemii Remizov, Ignacio Tejeda, Jarod Alper, Giovanni Inchiostro, and Vasily Ilin. Semantic search over 9 million mathematical theorems. *arXiv preprint*, 2026. URL https://arxiv.org/abs/2602.05216.

Anthropic. Formalizing Fermat's Last Theorem. https://www.anthropic.com/research/formalizing-fermats-last-theorem, 2026. Research post, September 4, 2026. Proof repository: https://github.com/anthropics/fermats-last-theorem.

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.

Andrej Bauer, Matej Petković, and Ljupčo Todorovski. MLFMF: Data sets for machine learning for mathematical formalization, 2023. URL https://arxiv.org/abs/2310.16005.

Carl Bergstrom. Eigenfactor: Measuring the value and prestige of scholarly journals. *College and Research Libraries News*, 68(5):314–316, 2007. ISSN 2150-6698. doi: 10.5860/crln.68.5.7804. URL https://crln.acrl.org/index.php/crlnews/article/view/7804.

Alex Best, Christopher Birkbeck, Riccardo Brasca, Eric Rodriguez Boidi, Ruben van de Velde, and Andrew Yang. A complete formalization of Fermat's Last Theorem for regular primes in Lean. *Annals of Formalized Mathematics*, 1:103–132, 2025. doi: 10.46298/afm.14586.

Rong Bian, Yu Geng, Zijian Yang, and Bing Cheng. AutoMathKG: The automated mathematical knowledge graph based on LLM and vector database, 2025. URL https://arxiv.org/abs/2505.13406.

Kevin Buzzard and contributors. Fermat's Last Theorem: Lean formalization blueprint. https://imperialcollegelondon.github.io/FLT/blueprint.pdf, 2026. Accessed 2026-09-12.

Jiangjie Chen, Wenxiang Chen, Jiacheng Du, Jinyi Hu, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Wenlei Shi, Zhihong Wang, Mingxuan Wang, Chenrui Wei, Shufa Wei, Huajian Xin, Fan Yang, Weihao Gao, Zheng Yuan, Tianyang Zhan, Zeyu Zheng, Tianxi Zhou, and Thomas Hanwen Zhu. Seed-Prover 1.5: Mastering undergraduate-level theorem proving via learning from experience, 2025. URL https://arxiv.org/abs/2512.17260.

Shuze Chen, Kunal Marwaha, Xiaoyang Lu, Henry Yuen, and Tianyi Peng. Prove2Me: An open collaborative platform for scaling math formalization, 2026. URL https://arxiv.org/abs/2608.28433.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pp. 625–635. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37.

frenzymath. jixia: A static analysis tool for Lean 4. https://github.com/frenzymath/jixia, 2024. GitHub repository, accessed 2026-05-23.

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*, pp. 8001–8013, 2024. URL https://aclanthology.org/2024.findings-emnlp.470/.

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*, 2025.

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, 2026. URL https://arxiv.org/abs/2605.13137.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*, 2022. URL https://arxiv.org/abs/2212.10496.

Eugene Garfield. Citation analysis as a tool in journal evaluation. *Science*, 178(4060):471–479, 1972. doi: 10.1126/science.178.4060.471. URL https://www.science.org/doi/abs/10.1126/science.178.4060.471.

J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, 102(46):16569–16572, 2005. doi: 10.1073/pnas.0507655102. URL https://www.pnas.org/doi/abs/10.1073/pnas.0507655102.

Lean Prover Community. doc-gen4: Document generator for Lean 4, 2026. URL https://github.com/leanprover/doc-gen4. Accessed May 25, 2026.

Xinze Li, Nanyun Peng, Simone Severini, and Patrick Shafto. The network structure of Mathlib, 2026. URL https://arxiv.org/abs/2604.24797.

Junqi Liu, Zihao Zhou, Zekai Zhu, Marco Dos Santos, Weikun He, Jiawei Liu, Ran Wang, Yunzhou Xie, Junqiao Zhao, Qiufeng Wang, Lihong Zhi, Jia Li, and Wenda Li. Numina-Lean-Agent: An open and general agentic reasoning system for formal mathematics, 2026. URL https://arxiv.org/abs/2601.14027.

Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2018. URL https://arxiv.org/abs/1603.09320.

Behrooz Mansouri, Vít Novotný, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. Overview of ARQMath-3 (2022): Third CLEF lab on answer retrieval for questions on math. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 13th International Conference of the CLEF Association, CLEF 2022, Bologna, Italy, September 5–8, 2022, Proceedings*, pp. 286–310, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-13642-9. doi: 10.1007/978-3-031-13643-6_20. URL https://doi.org/10.1007/978-3-031-13643-6_20.

Patrick Massot. leanblueprint: A plasTeX plugin for writing blueprints of Lean formalization projects. https://github.com/PatrickMassot/leanblueprint, 2020. GitHub repository; originally developed for the Sphere Eversion Project. Accessed 2026-05-26.

Shrey Mishra, Yacine Brihmouche, Théo Delemazure, Antoine Gauquier, and Pierre Senellart. First steps in building a knowledge base of mathematical results. In *Proceedings of the Fourth Workshop on Scholarly Document Processing*, 2024. URL https://aclanthology.org/2024.sdp-1.16.

Azim Ospanov, Farzan Farnia, and Roozbeh Yousefzadeh. miniF2F-Lean revisited: Reviewing limitations and charting a path forward, 2025. URL https://arxiv.org/abs/2511.03108.

Gabriel Pinski and Francis Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing and Management*, 12(5):297–312, 1976. ISSN 0306-4573. doi: https://doi.org/10.1016/0306-4573(76)90048-0. URL https://www.sciencedirect.com/science/article/pii/0306457376900480.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Delip Rao, Jonathan Young, Thomas Dietterich, and Chris Callison-Burch. WithdrarXiv: A large-scale dataset for retraction study, 2024. URL https://arxiv.org/abs/2412.03775.

Borja Requena, Austin Letson, Krystian Nowakowski, Izan Beltran-Ferreiro, and Leopoldo Sarra. A minimal agent for automated theorem proving, 2026. URL https://arxiv.org/abs/2602.24273.

Christian Steinfeldt and Helena Mihaljević. Evaluation and domain adaptation of similarity models for short mathematical texts. In *Intelligent Computer Mathematics: 17th International Conference, CICM 2024, Montréal, QC, Canada, August 5–9, 2024, Proceedings*, pp. 241–260, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-66996-5. doi: 10.1007/978-3-031-66997-2_14. URL https://doi.org/10.1007/978-3-031-66997-2_14.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, pp. 367–381, 2020. doi: 10.1145/3372885.3373824.

The mathlib Community. Mathlib statistics. https://leanprover-community.github.io/mathlib_stats.html, 2026. 137,089 definitions and 289,215 theorems. Accessed 2026-09-26.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition, 2024. URL https://arxiv.org/abs/2407.11214.

Dylan Walker, Huafeng Xie, Koon-Kiu Yan, and Sergei Maslov. Ranking scientific publications using a model of network traffic. *Journal of Statistical Mechanics: Theory and Experiment*, 2007 (06):P06010–P06010, 2007. ISSN 1742-5468. doi: 10.1088/1742-5468/2007/06/p06010. URL http://dx.doi.org/10.1088/1742-5468/2007/06/P06010.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. Naturalproofs: Mathematical theorem proving in natural language, 2021. URL https://arxiv.org/abs/2104.01112.

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, 2022.

Banri Yanahama and Akiyoshi Sannai. Lean atlas: An integrated proof environment for scalable human-ai collaborative formalization, 2026. URL https://arxiv.org/abs/2604.16347.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models, 2023. URL https://arxiv.org/abs/2306.15626.

Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Tompa. The tangent search engine: Improved similarity metrics and scalability for math formula search, 2015. URL https://arxiv.org/abs/1507.06235.

Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topić, and Kenny Davila. Ntcir-12 mathir task overview. In *NTCIR Conference on Evaluation of Information Access Technologies*, 2016. URL https://api.semanticscholar.org/CorpusID:9102694.

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, 2025. URL https://arxiv.org/abs/2506.05176.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics, 2022. URL https://arxiv.org/abs/2109.00110.

Karol Życzkowski. Citation graph, weighted impact factors and performance indices. *Scientometrics*, 85(1):301–315, 2010. doi: 10.1007/s11192-010-0208-6.

A Extended related work

Mathematical information retrieval. Early Math IR focused on formula-aware and keyword search, motivated by the fact that mathematical notation cannot be treated as ordinary text. Benchmarks such as NTCIR-11 Math-2, NTCIR-12 MathIR, and ARQMath formalized retrieval over mathematical corpora containing both formulas and surrounding text (Aizawa et al., 2014; Zanibbi et al., 2016; Mansouri et al., 2022), motivating representations such as Symbol Layout Trees and Operator Trees (Zanibbi et al., 2015). More recent work explores learned representations: zbMath-BERT studies embeddings for mathematical text (Steinfeldt & Mihaljević, 2024), NaturalProofs and TheoremSearch apply neural retrieval to informal statements (Welleck et al., 2021; Alexander et al., 2026), and LeanSearch retrieves Mathlib declarations from informal queries via sloganized formal statements (Gao et al., 2024; 2026). Our work extends sloganized statement retrieval by incorporating explicit dependency structure among mathematical statements.

Bibliometrics. Citation graphs model relationships among papers, authors, and journals; citation-count measures such as the h-index and journal impact factor summarize influence directly, while recursive prestige models weight citations by graph structure, recency, or linking behavior (Garfield, 1972; Hirsch, 2005; Pinski & Narin, 1976; Bergstrom, 2007; Walker et al., 2007; Życzkowski, 2010). In mathematics, graph-based analyses have been applied to Mathlib dependency structure and proof-assistant libraries as heterogeneous networks (Li et al., 2026; Bauer et al., 2023), while TheoremKB and AutoMathKG construct theorem-level graphs for informal sources at smaller scale (Mishra et al., 2024; Bian et al., 2025). We extend these graph-based perspectives to theorem-level dependencies across both informal and formal mathematics.

Neural theorem proving and autoformalization. Autoformalization translates informal statements into formal declarations, while neural theorem proving generates proofs for those declarations. Progress on benchmarks such as miniF2F and PutnamBench has increased the importance of retrieval and tooling for agentic proving systems (Zheng et al., 2022; Ospanov et al., 2025; Tsoukalas et al., 2024; Requena et al., 2026; Liu et al., 2026). Our work supports this direction by aligning informal and formal statements with their dependency context.

Lean dependency graphs. Dependency extraction for formal libraries is well studied (Alama et al., 2012), and several Lean tools build declaration-level graphs. Jixia, the extractor behind LeanSearch v2 (frenzymath, 2024), emits a raw post-elaboration constant-reference graph; Lean Atlas (Yanahama & Sannai, 2026) splits type- and value-level edges and adds confidence scores, `sorry`-status, reachability, and visualization; and `leanblueprint` (Massot, 2020) provides author-annotated links between informal LaTeX statements and their formal Lean declarations. LeanGraph is complementary: it classifies dependencies through Lean's kernel APIs, filters kernel artifacts with a documentation-aligned node set, and refines prior edge taxonomies into `sig`/`extends`/`field` and `proof`/`def` edges at multi-library scale.

B In-context literature search

Research mathematicians often need to identify a known result from a partial description embedded in an ongoing proof. We study a related agentic task: recovering a cited theorem from a local passage of mathematical text after its citation has been removed.

We sampled 149 passages from mathematics arXiv papers released after February 2026 (to reduce pretraining contamination), each containing a citation to a named theorem-like statement in another arXiv paper. We replaced the citation with `[Citation Needed]` and removed the corresponding entry from the paper's bibliography. We evaluate three settings: unrestricted *web search*; *dense retrieval*, where models issue repeated searches against our statement database receiving top-10 candidates per query; and *graph-based retrieval*, where models additionally navigate the statement dependency graph. *Exact match* requires the correct theorem identifier and cited arXiv identifier; *paper-level match* requires only the correct paper.

Graph-based retrieval improves exact-match performance over dense retrieval for all models, and paper-level retrieval for two of three (Table 5), so graph navigation provides useful structure beyond

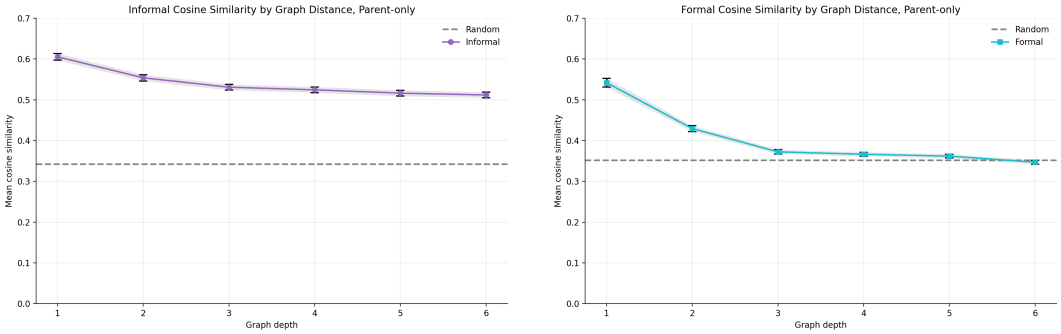
Table 5: In-context citation recovery on 149 redacted arXiv passages.

System	Exact	Paper-level
Web search baseline		
ChatGPT 5.4	0.336	0.523
Gemini 3.1 Pro	**0.570**	**0.752**
Claude Sonnet 4.5	0.181	0.443
Dense retrieval		
ChatGPT 5.4	0.228	0.409
Gemini 3.1 Pro	0.181	0.409
Claude Sonnet 4.5	0.121	0.396
Graph-based retrieval		
ChatGPT 5.4	0.255	0.456
Gemini 3.1 Pro	0.289	0.597
Claude Sonnet 4.5	0.148	0.322

dense statement retrieval. Neither database-backed method outperforms unrestricted web search—though the web baseline is not a controlled comparison, since models can recover the original source paper by searching an exact quotation from the passage, partly measuring access to the evaluation instance itself. The experiment also permits repeated retrieval actions without a matched search budget across conditions, and the 149 examples cover only citations that explicitly name a theorem-like result.

C Node distance and embedding distance

We examine whether statements closer in the dependency graph have more similar slogan embeddings, measuring cosine similarity between pairs of statements separated by directed parent-only graph distance d. For each formality and depth, we collected 1,000 accepted statement pairs by random parent-following walks from random seeds, retaining a pair only when its directed shortest-path distance is exactly d. A dashed baseline gives the mean similarity of 1,000 random distinct pairs per corpus.



(a) Informal cosine similarity by d-hop distance.

(b) Formal cosine similarity by d-hop distance.

Figure 4: Cosine similarity and node depth for informal and formal graphs.

Similarity declines with distance in both graphs, but far more sharply in the formal graph: from $\sim$0.54 at depth one to roughly the random baseline by depth six, whereas informal pairs decline only from $\sim$0.61 to $\sim$0.51, well above baseline (Figure 4). A plausible explanation is the formal graph's much higher local connectivity: Lean records fine-grained dependencies on definitions, typeclass instances, and reusable infrastructure, so short formal paths quickly reach broadly applicable declarations that share little high-level semantic content with the seed, while informal dependency paths more often remain within a shared local development. This analysis conditions on successful walks with embedding-eligible endpoints, so samples at a given depth may favor statements with rich local neighborhoods.

D LeanSearch v2 comparison details

Reranking protocol. We rerank with Qwen3-Reranker-8B using LSv2's prompt and its relevance score, $P(\text{yes})/(P(\text{yes}) + P(\text{no}))$ from the reranker's final token, and rank by that score alone. We report two depths: top-50, as described in the LSv2 paper, and top-200, as in its released code. The reranked rows rerank F's saved 200-candidate pool, whose retriever-only score (0.738 Recall@10, 0.562 nDCG@10) is within 0.5pp of the F row. Candidates are turned into LSv2-style passages from our corpus, so this is not a byte-identical copy of the LSv2 corpus.

Reproducing LSv2. We ran LSv2's pipeline ourselves: its released corpus of 310,579 passages, its query format, and its reranking prompt, executed with our own code (Table 6). After reranking, every metric is within 0.3pp of the reported value. Before reranking, our Recall@10 is 1.1pp higher, plausibly because we search exactly where LSv2 uses an approximate index.

	nDCG@1	nDCG@5	nDCG@10	Recall@10
LSv2 retriever, reported	0.340	0.472	0.494	0.657
LSv2 retriever, our run	0.342	0.470	0.498	0.668
LSv2 reranked, reported	0.470	0.601	0.623	0.780
LSv2 reranked, our run	0.467	0.599	0.621	0.780

Table 6: Our run of LSv2's released pipeline against its reported numbers (MathlibQR fair-810).

Confidence intervals. Table 7 gives 95% bootstrap intervals, resampling the 171 target declarations with all their queries (5,000 resamples). The lower ends of both reranked configurations lie above LSv2's reported reranked scores. E's interval contains LSv2's reranked recall of 0.780, so for E we find no difference rather than proving equality.

	Recall@10	nDCG@10
E (no reranker)	0.775 [0.730, 0.820]	0.548 [0.502, 0.593]
F + reranker, top-50	0.843 [0.810, 0.875]	0.715 [0.679, 0.751]
F + reranker, top-200	0.883 [0.853, 0.911]	0.739 [0.704, 0.772]
LSv2 reranked, reported	0.780	0.623

Table 7: 95% bootstrap intervals on MathlibQR fair-810.

Ablation details. D, which embeds both passages, gains 3.4pp Recall@10 over F but loses 4.8pp nDCG@1 (0.343 vs. 0.391). E adds 18.9pp Recall@10 and 16.8pp nDCG@10 over the baseline A.

E Limitations

TheoremGraph is built from arXiv LaTeX source, a large structured corpus that does not cover all mathematical writing. Informal dependency extraction is necessarily approximate, so the released graph preserves per-extractor labels, letting users pick their own precision–coverage tradeoff.

The cross-formality bridge relies on LLM-generated slogans and Qwen3-Embedding-8B embeddings, and sloganization is lossy. The matching judge works from slogans with limited context, and our expert calibration covers only ten pairs, so not every judge-affirmed match is ground truth. Giving judges more context, such as the source paper, is a natural way to improve cross-corpus linking, alongside evaluating alternative slogan generators, embedding models, rerankers, and a broader judge panel. Because matching takes only the single nearest candidate over an arXiv-only corpus, well-known results go unmatched when their counterpart is absent.

Our downstream evaluations focus on settings with gold labels: blueprint pairs, 24 autoformalization targets, and the MathlibQR fair-810 subset. The retrieval-side settings tuned for MathlibQR do not transfer to chained-premise retrieval: applying them lowers MathlibMPR group Recall@10 from 0.224 to 0.165, in a retriever-only comparison. We have not evaluated the reranked system on

MathlibMPR. Concept retrieval and chained-premise retrieval appear to need different signals, which we leave to future work.

Data availability. The TheoremGraph dataset, the LeanGraph and informal extractors, and the HTTP API and MCP interface are publicly available (links withheld for anonymous review). Because arXiv's default license grants no third-party redistribution right, the public release of judged matches is restricted to statements carrying an open, redistributable license: a subset of 23,399 judged candidates, gated for non-commercial research, with our own contributions (slogans, judgments, metadata) under CC-BY-NC-SA-4.0 and each record carrying its statement's original license in a `source_license` field.