Qeios

Peer Review

Review of: "LemmaHead: RAG Assisted Proof Generation Using Large Language Models"

Evgenii Arsentev[1]

1. Independent researcher

What the paper claims

LemmaHead is a retrieval base built from Olympiad textbooks, transcribed page by page into LaTeX with a vision model, chunked so that a problem stays with its solution, embedded and stored in Chroma (Section 4.2). It is placed in front of GPT-4 in three pipelines: plain retrieval; retrieval with enhanced query generation, where the model first lists the concepts and lemmas it thinks it needs; and iterative proof augmentation, which repeats the query-and-redraft loop for sigma = 5 rounds before the informal proof is translated into Lean and checked (Sections 4.3.1 and 4.3.2). Evaluation is Pass@1 on miniF2F, 488 problems split equally into validation and test (Section 4.1). Table 1 (validation) reports GPT-4 at 9.4%, plain RAG at 2.3%, enhanced queries at 25.2%, the iterative pipeline at 40.0%, against 11.5% for human-guided GPT-4 and 23.9% for GPT-f; Table 2 (test) reports 9.0%, 2.5%, 27.6%, 32.4%, 8.6%, and 24.6%.

The idea is sensible, and the negative result inside it is more interesting than the positive one. I will come back to that. The problems below are mostly about what the tables are allowed to support.

Does the conclusion hold on the data?

1. The iterative pipeline is not Pass@1 in the sense the baselines are. Section 4.3.2 runs five rounds of query generation and proof redrafting before a formal proof is emitted, so one reported attempt costs at least five model calls plus retrievals, while the GPT-4 baseline in the same column costs one. Comparing them under a single label makes a budget difference look like a method difference. The comparison a reader needs is compute-matched: plain GPT-4 with five samples scored by the same Lean verifier, or every method reported with calls per problem alongside the success rate. Until that is done, the headline gain cannot be separated from simply sampling five times.

doi.org/10.32388/59NX5L

2. The state-of-the-art numbers are borrowed and not like-for-like. The last two columns of both tables come from other papers, cited as [6] and [7], and were produced with different models, different proof search budgets, and different Lean and mathlib versions than the ones used here. On that basis, the claim in Section 6 that the method "substantially outperform[s] state-of-the-art proof generators" is not supported. Either re-run at least one of those baselines in your own environment or restate the claim as a comparison against your own GPT-4 baseline, which is the only controlled contrast in the paper.

3. The differences are inside the noise of a 244-problem split. Each split holds 244 problems, so the standard error of a rate near 30% is about 2.9 percentage points, and the 95% interval on a difference between two such rates is roughly eight points. On the test set, enhanced queries at 27.6% against GPT-f at 24.6% is a three-point gap, well inside that band; the iterative pipeline at 32.4% against 24.6% is 7.8 points, right at its edge. No seeds, repetitions, or intervals are reported anywhere, and GPT-4 sampling is stochastic. Repeat each pipeline at least three times, report the mean and standard deviation, and mark which comparisons survive.

4. The gap between validation and test for the iterative pipeline is large and unexplained. The same method scores 40.0% on validation (Table 1) and 32.4% on test (Table 2), a drop of 7.6 points, while enhanced queries move the other way, 25.2% to 27.6%. If sigma, the chunking, the prompts, or the top-k were chosen while looking at validation results, that is selection on the validation split and should be stated. Say explicitly what was tuned, on which split, and how many configurations were tried.

5. The most valuable result in the paper is the one that is waved away. Plain retrieval does not merely fail to help; it destroys the baseline: 9.4% to 2.3% on validation and 9.0% to 2.5% on test, a four-fold drop. Section 5 attributes this to "poor querying" in a single sentence. That is a hypothesis, and it is testable with the material already in hand. Measure retrieval quality directly against the lemmas a problem actually needs, and run the obvious control where the retrieved passages are replaced by random chunks from the same corpus. If random context is no worse than retrieved context, the retrieval stage is not merely weak; it is noise. A paper that established that cleanly would be more useful than one that reports a higher Pass@1.

Reproducibility

This is the section that needs the most work, and for this particular paper, the missing items are not cosmetic, because a proof either compiles or it does not, and whether it compiles depends on the toolchain.

- No Lean version and no mathlib commit are given. The same proof script passes under one mathlib and fails under another, so every number in Tables 1 and 2 is currently unverifiable. State both, and say whether the Lean 3 or the Lean 4 version of miniF2F was used.
- The GPT-4 snapshot is not named, and no decoding parameters are given - temperature, top-p, maximum tokens, seed. For a single-attempt success rate, these are the difference between a reproducible number and an anecdote.
- Section 4.2.3 says chunks were embedded "through GPT-based embedding functions" without naming the embedding model, the chunk size, the overlap, or the top-k passages returned at inference. Retrieval behavior is entirely determined by those four choices.
- Section 4.1 does not say whether the model was given the formal Lean theorem statement or had to produce it as well, nor whether generated proofs were screened for degenerate closures such as sorry or trivial tactics. Both change what Pass@1 means.
- No code, no knowledge base, no list of the textbooks used. The corpus is the contribution named in the title, so its absence is the main obstacle to anyone building on this.
- Section 4.2.1 describes transcribing published textbooks into LaTeX with a vision model. The licensing status of that derived corpus is not addressed anywhere, and it determines whether the knowledge base can be shared at all. Please state it.

What to fix

1. Section 5 and Tables 1 and 2: report calls per problem for every pipeline, and add a compute-matched baseline, GPT-4 with five samples verified the same way, so the iterative gain is separated from the sampling budget.
2. Section 5: either reproduce the borrowed baselines in your own environment or restate the claim in Section 6 as a comparison against your own GPT-4 baseline. Also, give the precise source of each borrowed number, by table and row.
3. Sections 5 and 6: repeat each configuration at least three times and report the mean, standard deviation, and the interval on each difference. At 244 problems per split, gaps below roughly eight points should not be described as outperformance.
4. Section 4.3.2: sigma is described as "arbitrarily set" to 5, and Section 6 says rates "seem to increase with more iterations." Plot Pass@1 against iteration 1 to 5 from the runs you already have; that curve is the evidence for the claim, and it costs nothing to print.

5. Section 5: investigate the plain-RAG collapse. Report retrieval precision against the lemmas required, and add the random-context control.
6. Section 4.1: state whether the formal statement was provided, and how degenerate proofs were excluded.
7. Section 4.2: name the embedding model, chunk size, overlap, and top-k; give the Lean and mathlib versions and the GPT-4 snapshot with decoding parameters.
8. Section 4.2.1: address the licensing of the transcribed textbook corpus, and release what can be released.
9. Section 3: the related work is a sequence of one-sentence summaries with no synthesis, and it contains several typos - "highlightened," "theoritical," "reseach." Section 4.3 has "the the generated formal proof"; the abstract has "LLMS." A careful pass is needed.
10. Front matter: the paper lists affiliations by number but no institution names, and correspondence is routed through the publisher rather than an author address. Add both.

Recommendation

Major revisions. The construction is reasonable, and the pipeline is described clearly enough to follow, but as it stands, the central comparison is between systems with different compute budgets, against baselines lifted from other papers and other toolchains, on differences a 244-problem split cannot resolve, with no Lean version that would let anyone check a single proof. The good news is that the two most valuable additions are cheap: a compute-matched baseline, and an honest investigation of why retrieval made things four times worse. The second one is, in my view, the paper you actually have.

Evgenii Arsentev, PhD

Declarations

Potential competing interests: No potential competing interests to declare.