Extended Abstract

Motivation Formalizing mathematics in Lean 4 de Moura et al. (2015) has gained significant momentum in recent years. Mathlib mathlib Community (2020), Lean's central mathematical library, now contains over one hundred thousand formalized results spanning areas such as combinatorics, number theory, and probability. A central problem in this area and automated theorem proving in general is finding the right theorems, lemmas, and definitions needed to prove a goal–a task known as premise retrieval. The goal of this project is to see whether we can use reinforcement learning to distill premise retrieval skills from larger models. In our context, "larger model" means either an LLM with more parameters or a strong automated theorem prover such as Harmonic's Aristotle Achim et al. (2025).

Method We build on the LeanSearch-style baseline Gao et al. (2026), which embeds Mathlib statements with Qwen-8B and retrieves candidates by cosine similarity. To create supervision, we scrape completed Mathlib proofs, replace proof fragments with "sorry," and record the resulting Lean state, goal, natural language translation, and theorem used to finish the proof. We also generate a smaller distillation dataset by asking Aristotle Achim et al. (2025) to prove sorry-replaced measure theory theorems and recording the premises used in successful proofs. We then post-train an embedding model and train a reranker model with this data.

Implementation We train on two datasets: roughly 20k Mathlib examples with a single ground-truth theorem and roughly 3k measure-theory examples with Aristotle-generated premise lists. The baseline returns the top 100 candidates from the embedding index. As an attempt at distillation, we tried to fine-tune the smaller Qwen/Qwen3-Embedding-0.6B model (through LoRA weights) with REINFORCE + baseline, using rewards based on the ground truth dot product using the original 8B Qwen model embeddings. For another avenue, a reranker network scores the top 100 candidates using the base Qwen8B. We first warm-start the reranker with supervised fine-tuning using cross-entropy on the ground-truth index, then apply REINFORCE with Plackett–Luce sampling, a greedy baseline advantage estimate, an nDCG@10 reward, and a KL penalty to keep the policy close to the supervised model.

Results The distillation fine-tuning failed: when the fine-tuned 0.6B Qwen model was used to rank theorems by cosine similarity, Hit@100 was zero across the training set. On the other hand, supervised reranking consistently improved over the embedding baseline. On the Mathlib dataset, nDCG@10 improved from 0.320 to 0.399, Hit@5 from 0.434 to 0.503, and MRR@100 from 0.255 to 0.336. Adding RL reduced these scores slightly to 0.389, 0.485, and 0.327. On the Aristotle measure-theory dataset, SFT improved nDCG@10 from 0.191 to 0.236, Hit@5 from 0.410 to 0.495, and MRR@100 from 0.132 to 0.163; SFT+RL gave 0.233, 0.505, and 0.165.

Discussion In fine-tuning, the training curves stayed mostly flat, with no indication of real model improvement (even on the training set itself) over 10 epochs on the 20k–25k context–goal–ground-truth dataset. In reranking, most gains came from supervised fine-tuning rather than reinforcement learning. Pure RL tended to overfit, motivating the KL penalty, and even with this regularization the Mathlib results degraded slightly after RL. The Aristotle setting was more encouraging: RL produced modest gains on Hit@5 and MRR@100, suggesting that prover-generated multi-premise rewards may provide a useful signal. Training curves also showed SFT overfitting after about five epochs, indicating that the current datasets are not yet rich enough.

Conclusion Lightweight reranking can substantially improve Lean premise retrieval over a cosine-similarity embedding baseline. RL alone did not reliably improve the reranker, but distillation from Aristotle appears promising. Furthermore, an attempt at also fine-tuning the 8B model directly and avoiding the distillation goal altogether may help improve at the premise retrieval task, even more so when combined with a reranker. Future progress likely depends on larger and more diverse generated datasets and as well as exploring hierarchical RL methods.

Improving Lean Premise Retrieval via RL and Distillation

Alexander López
Department of Mathematics
Stanford University
ald39@stanford.edu

Fred Rajasekaran
Department of Mathematics
Stanford University
fredr@stanford.edu

Kevin Rizk
Department of Mathematics
Stanford University
krizk@stanford.edu

Abstract

Premise retrieval is a central bottleneck in automated theorem proving and autoformalization in Lean: given a proof goal, a system must identify the Mathlib theorems, lemmas, and definitions most likely to complete the proof. We investigate whether reinforcement learning and distillation from stronger theorem provers can improve Lean premise retrieval. We build a data-generation pipeline that scrapes completed Mathlib proofs, replaces proof fragments with `sorry`, and records the resulting Lean context, goal, natural language translation, and ground truth theorem. We also generate a smaller Measure Theory dataset by asking Aristotle to prove `sorry`-replaced theorems and recording the premises used in successful proofs. We fine-tune a smaller (0.6B parameter) embedding model for the premise retrieval task through REINFORCE + baseline using rewards from a larger (8B parameter) model. We also train a reranker over the top 100 retrieved candidates, first with supervised fine-tuning and then with REINFORCE using an nDCG@10 reward and a KL penalty. Supervised reranking accounts for most of the improvement. Adding RL gives mixed results, as it performs slightly worse on the Mathlib data set but better on some of the Aristotle data set metrics. Overall, our results suggest that lightweight supervised reranking is effective for Lean premise retrieval, and RL gives mixed results, with no successful distillation at the embedding stage. The main bottleneck is most likely due to not having enough data.

1 Introduction

Formalizing mathematics in Lean 4 de Moura et al. (2015) has gained significant momentum in recent years. Mathlib mathlib Community (2020), Lean's central mathematical library, now contains over one hundred thousand formalized results spanning areas such as combinatorics, number theory, and probability.

The use of artificial intelligence for autoformalization in Lean has also attracted increasing attention. A major remaining challenge is premise retrieval: given a mathematical goal, the task is to identify the Mathlib theorems and definitions that are most likely to be useful in proving it. Effective premise retrieval is valuable both for automated formalization pipelines and for mathematicians independently formalizing results in Lean.

In this work, we investigate whether reinforcement learning can improve premise retrieval for Lean. We develop a method for generating large-scale training data from Mathlib and fine-tune an embedding model using REINFORCE. The latter consists of testing whether a fine-tuned embedding of a smaller model with learning signal coming from a significantly larger model helps to transfer ability in the premise retrieval task. We also study reinforcement-learning-based reranking methods and evaluate how they improve over baseline retrieval approaches. Finally, we examine whether

distilling retrieval signals from strong Lean automated theorem provers leads to further performance gains.

Our main contributions are as follows:

- Developed a data pipeline for generating training data from Mathlib.
- Post-trained an embedding model.
- Trained a reranker using reinforcement learning.

2 Related Work

A lot of recent work has focused on premise retrieval. Yang et al. (2023) introduced ReProver, an LLM-based prover augmented with retrieval for selecting premises from a large mathematical library. Subsequent work has built on this idea in several directions: Tao et al. (2025) improved negative mining; Shen et al. (2025) integrated retrieval into end-to-end provers; and Petrovčič et al. (2025) used graph-augmented embeddings. More recently, Gao et al. (2026) introduced a reasoning mode based on an iterative sketch-retrieve-reflect pipeline for generating a list of relevant theorems.

Given our limited resources, we aimed to develop a fast and less computationally intensive premise-retrieval method compared to previous work. Therefore, we primarily focused on and built upon the search-retrieval approach of Gao et al. (2026), which embeds the Mathlib corpus into a high-dimensional vector space and, given a query, retrieves the most relevant theorems using cosine similarity.

3 Method

3.1 Data Generation

One major challenge is obtaining a large amount of high-quality training data. We developed a pipeline to extract such data from Mathlib, which we describe below. Each data point consists of a Lean state, including all variables and hypotheses; a Lean goal, namely what remains to be proved; their natural-language translations; and the ground truth: a list of Mathlib theorems and statements that could be used to complete the Lean goal.

We generate two different datasets: one obtained by scraping Mathlib, described in Section 3.1.1, and another constructed using Aristotle Achim et al. (2025), described in Section 3.1.2.

3.1.1 Scraping Mathlib

We scrape Mathlib, Lean's math library, for examples of completed Lean proofs. We randomly "sorry" them, and then store the broken proof. At the point where we included a "sorry", we scrape the current Lean state (all variables and hypotheses), and the current Lean goal (what remains to be proved), which we will call the "context". We also scrape the resulting Mathlib theorems that were used to finish the proof. The theorems were selected at random within a proof using a Python interface for the Lean server PyPantograph [Aniva et al. (2024)]. This has the side effect of not selecting 'key' theorems within a specific proof, and a non-trivial proportion of the examples include basic tactic applications that are common throughout Mathlib, and don't carry as much information about the current proof strategy.

After extracting the tuples of (context, goal, theorem) from many Mathlib proofs, we used DeepSeek's deepseek-v4-flash (thinking enabled) to provide natural language translations for the Lean context, theorem and goal, avoiding as much Lean syntax as possible, and expressing any necessary mathematical equations in LaTeX, or with direct natural language instead.

3.1.2 Data generation through Aristotle

We use a powerful automated theorem prover, Aristotle Achim et al. (2025), to generate ground-truth labels for premise retrieval. This also allows us to obtain Lean proofs that differ from the original Mathlib proofs. For each theorem, we construct a Lean file in which the theorem body is replaced by 'sorry', as described in the previous section, and then ask Aristotle to synthesize a proof. When

Aristotle succeeds, we record the Mathlib theorems invoked in the generated proof. These premises, together with the original Lean goal and local Lean context, form a labeled training data point.

Since this pipeline is computationally expensive, we focus primarily on extracting data points from measure theory, using this domain as a case study to evaluate whether data generated by an automated theorem prover can improve premise retrieval performance.

3.2 Baseline Method

For the baseline method, we followed the LeanSearch Gao et al. (2026) approach. We used their embeddings of the Mathlib corpus, generated with the Qwen-8B model. Given a query, we returned the top-k closest results based on cosine similarity.

3.3 Embedding improvement through REINFORCE

We also used REINFORCE + baseline to fine-tune the Qwen-Embedding-0.6B model with a LoRA adapter. The particular strategy involved using the embeddings from the Qwen-Embedding-8B model obtained by LeanSearch (by embedding their Mathlib theorem/declaration corpus) as a source of rewards in a pooled multi-armed bandit setting.

The training was as follows: one copy of the fine-tuned model would be kept fixed for some fixed number of batched gradient steps. The gradient steps consisted of using the fixed copy of the fine-tuned model to embed a given (context, goal) tuple and use these embeddings to pick a pool of the top k candidates from the Mathlib theorem/declaration corpus. The pool is then augmented with the ground truth theorem belonging to the current training example, and a selection of r other random Mathlib theorems. Then the up to date fine-tuned model would sample from a categorical distribution on this smaller pool with logits corresponding to the dot product (related to cosine-similarity) between the embedded query and the selected embedded theorems (these embeddings come from the up to date fine-tuned model). Finally, the result was rewarded according to the dot product of the ground truth and the selected theorem using embeddings coming from the 8B parameter model (fixed, with no additional weights), plus a fixed scalar bonus if the ground truth and the selected theorem are the same.

3.4 Reranker Network

On top of updated the underlying embedding model as described above, we also trained a smaller neural network to act as a reranker model. Specifically, the LLM (either the baseline Qwen model or the post-trained one above) returns a list of 100 candidate theorems which are chosen via cosine similarity to the query. Then, the reranker network outputs a score for each of these 100 candidates. The higher the score, the more likely that the given theorem/lemma/definition will be relevant for completing the proof described in the query. We train the reranker network first via supervised fine-tuning and then with RL.

3.4.1 Supervised Fine-Tuning

The reranker network is given a warm start via supervised fine-tuning. Here, we use a cross-entropy loss to train the network. Specifically, we use cross entropy with the ground truth index so that the score assigned to the ground truth is higher.

3.4.2 RL via REINFORCE and a KL Penalty

Our experiments showed that most of the benefit came from the supervised fine-tuning, and the RL with pure REINFORCE was overfitting. Thus we added a KL penalty to the RL loss so that the training did not stray too far from the original SFT baseline. The reward we used for REINFORCE was nDCG@k with $k = 10$. Mathematically, if the ground truth is at rank r,

$$r_{nDCG} = \begin{cases} \frac{1}{\log_2(r+1)} & r \leq 10 \\ 0 & \text{otherwise.} \end{cases}$$

We also tested a reward system where we did not cut off the reward after the first 10 positions. However, this reward did not penalize the reranker model enough for ranking the ground truth in

higher positions since the reward was given as $\frac{1}{\log_2(r+1)}$ which does not decay fast as the rank tends to 100. We use REINFORCE with a baseline to train the model. The reranker model samples 10 actions from a categorical distribution whose logits are determined by the scores assigned by the model (Plackett-Luce sampling). It samples 10 theorems. It computes the baseline reward by getting reward of greedily ranking the candidates by their scores (i.e. highest score gets the first position, second highest score gets the second, etc.). It then estimates the advantage by computing the reward of the sampled theorems minus the greedy baseline. With this reward we do REINFORCE, so that

$$L_{nDCG} = -\frac{1}{N}\sum_{n=1}^{N} A_n(s,a)\log \pi_{\theta,n}(a|s),$$

where π_θ is the reranker model and the action is determined by the scores assigned to the candidates and the stochasticity from the sampling. Note that this is similar to a contextual-bandit style problem, and the state is determined by the embedding of the query. If π_{SFT} is the baseline model after the warm start, the total loss is

$$L = L_{nDCG} + 0.05 D_{KL}(\pi_\theta || \pi_{SFT}).$$

In the case when we had multiple ground truth theorems (as in the case where we had Aristotle rewards), we took the average of r_{nDCG} over each ground truth theorem.

4 Experimental Setup

4.1 Training Dataset

For the training dataset, we used two different datasets. The first contains about 20k examples generated as described in Section 3.1.1, where each ground-truth label contains only one Mathlib theorem. The second, described in Section 3.1.2, contains about 3k examples focused on measure theory, where each ground-truth label contains a list of relevant theorems.

4.2 Baseline: Context Representation

Before reranking, we need a sufficiently strong baseline method that retrieves many of the target theorems within the top 50 or 100 results. We tested different context representations by embedding them with Qwen-8B and retrieving the top-k closest results. Given a Lean context, a Lean goal, and their natural language translations, we evaluated whether different context representations yielded better results, measured by the frequency with which the target theorems appeared in the top-k retrieved results. Specifically, we considered the following query representations:

- Using only the Lean goal and context
- Using only the natural language translation of the Lean goal and context
- Using both the Lean representation and the natural language translation

4.3 Embedding via REINFORCE

We tested the distilled embedding using a method similar to the baseline. We embedded the Mathlib corpus with the post-trained Qwen-0.6B model. Then, for each test data point, we embedded the Lean context, goal, and natural language description, and retrieved the top-k Mathlib theorems using cosine similarity.

4.4 Reranker Network

We tested the reranker network in two separate experiments on the datasets described in Section 4.1. We first did SFT and then used the best checkpoint from SFT to start the RL. We only tested on data points where the ground truth was in the top 100 candidate theorems, which was around 53.3% of the queries as shown in Figure 1.

5 Results

5.1 Evaluation

5.1.1 Test Dataset

To ensure a fair comparison between our different approaches, we held out a portion of the data generated and used it as a test set.

5.1.2 Evaluation Metrics

To evaluate our approaches, we used several standard ranking metrics: Hit@K, MRR@K, and nDCG@K.

Hit@K measures whether the target theorem appears among the top K retrieved theorems, averaged over all data points:

$$\text{Hit@K} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{\text{rank}_i \leq K\},$$

where rank_i is the rank of the target theorem for the i-th query.

Mean Reciprocal Rank (MRR) and nDCG measure the quality of the ranking by taking into account the position of the relevant result. They assign higher scores when the correct item appears earlier in the ranked list. MRR@K is defined as:

$$\text{MRR@K} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{\text{rank}_i \leq K\}\frac{1}{\text{rank}_i},$$

and nDCG@K is defined as:

$$\text{nDCG@K} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{\text{rank}_i \leq K\}\frac{1}{\log_2(\text{rank}_i + 1)}.$$

5.2 Context Representation Baseline Method

As shown in Fig. 1, there is no significant difference in performance among the three context representations. However, we observe that using only natural language performs worse than the other two representations, with a difference between 1.3% and 2.5%. Furthermore, there is a high correlation among the data points that appear in the top-K: for example, if a data point is in the top 50 when using only natural language, there is also a high chance that the same data point appears in the top 50 when using the Lean representation.

Nonetheless, the best performance, with 53.3% of the relevant results appearing in the top-K, is achieved by using the natural-language and Lean contexts together, as expected. Therefore, we use this context representation in the rest of our experiments.

5.3 Reranker with Raw Mathlib Data and Aristotle Distillation

As evidenced in Figure 2, the supervised fine-tuning performed better than the reinforcement learning. The plot shows the reward we used $nDCG@10$ for training, Hit@k figures with shows the proportion of queries that had the ground truth theorem in the top k items, and MRR@100, which is the mean reciprocal rank of the ground truth theorem in the 100 candidates. The quantitative values are enlarged in Table 1.

6 Discussion

6.1 Issues with Dataset Generation

The main issue with the dataset generation is the way of theorem extraction from Mathlib. At the moment there is no simple way of using PyPantograph, or any other means known to the authors,

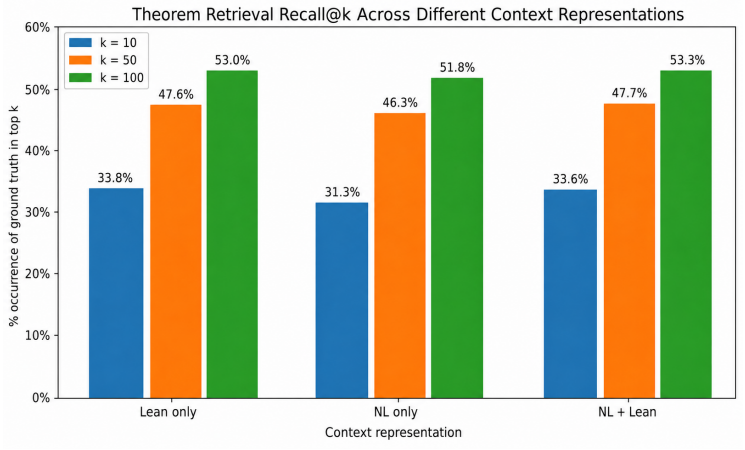

Figure 1: The percentage of appearances of the target theorem in the top-k retrieved results using different context representations

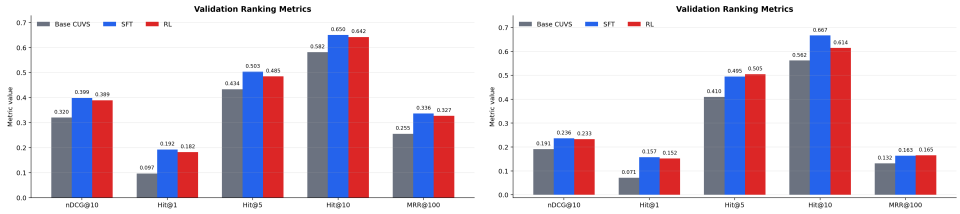

Figure 2: The performance of the SFT and RL network compared to the base embedding distances via cosine similarity. Left: dataset from Mathlib without Aristotle rewards. Right: measure theory dataset with extra Aristotle rewards.

Table 1: Performance Comparison for Mathlib and Aristotle Data

	Method	nDCG@10	Hit@5	MRR@100
Mathlib:	Baseline	0.320	0.434	0.255
	SFT	0.399	0.503	0.336
	SFT + RL	0.389	0.485	0.327
	Method	nDCG@10	Hit@5	MRR@100
Aristotle:	Baseline	0.191	0.410	0.132
	SFT	0.236	0.495	0.163
	SFT + RL	0.233	0.505	0.165

of distinguishing the relevance of a given Mathlib theorem within its proof context. The random selection implemented for our dataset generation makes the (context, goal, ground truth) very noisy, where the ground truth could sometimes involve a trivial, say 'commutativity of addition' theorem within a more elaborate proof, where the Lean tactic application also involved a more significant theorem.

Another issue, relating to the scale of the dataset, is the limitation of the natural language translation due to the cost of the DeepSeek translations. High quality translations required activating the thinking mode, even on the cheapest available model (deepseek-v4-flash), and translating a corpus on the scale of Mathlib was out of the scope for this project.

Aristotle Achim et al. (2025) as described in section 3.1.2, is very slow, and it takes a significant amount of time to prove and complete Lean files. In addition, for some of the theorems it was asked to complete, Aristotle recognized that the statement was already a Mathlib theorem with some parts replaced by sorry. As a result, it simply copied the original Mathlib proof, which defeated the purpose of generating diverse Lean proofs. When we forced Aristotle not to copy existing Mathlib proofs, the generation process became even slower and required substantially more time. One possible direction for addressing this issue is to use a different automated prover or to provide Aristotle with proof tasks in a more intelligent way.

6.2 Distilling Embeddings

Based on our results it would seem that distilling knowledge of the 8B parameter Qwen model into the 0.6B parameter model is ineffective via reinforcement learning (using 8B parameter informed rewards). Throughout training, the average reward curve remained flat even after multiple epochs on the same dataset (not even a sign of overfitting) as shown in Fig. 6.2

Figure 3: Flat mean reward learning curve for LoRA fine-tuning of the Qwen-Embedding-0.6B model.

Moreover, the main limitations for fine-tuning involved the pool selection. The pool size was quite limited due to a joint embedding of the pool and the (context, goal) queries. Having a larger pool restricted the batch size, and vice-versa, so using a single GPU restricted our training to a pool size of 6 (top 3 theorems + ground truth and two random theorems) and batch size 10.

6.3 Reranker Network

Across the board, SFT did most of the improvement compared to the base CUVS embeddings from LeanSearch. The reinforcement learning layer on top of the SFT only made marginal changes. For the Mathlib data, it actually negatively affected the performance on the validation set. This is because the RL was overfitting on the training data. For the Aristotle rewards system, however, there were

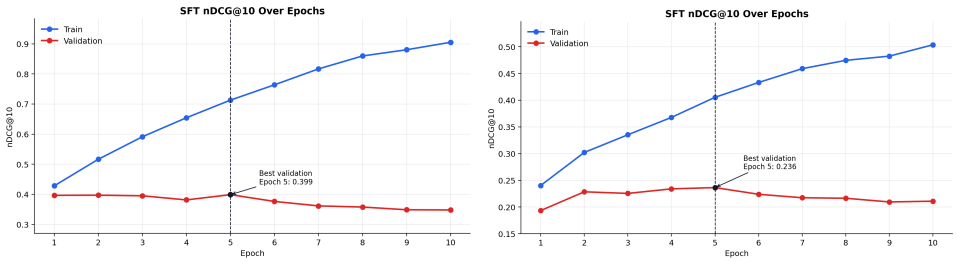

Figure 4: The training curves for SFT over 10 epochs with validation and test error. Left: dataset from Mathlib without Aristotle rewards. Right: measure theory dataset with extra Aristotle rewards.

improvements in the validation metrics that were not the actual training loss (nDCG@10). Thus, RL gave modest but positive rewards on the validation set in the distillation reward setting.

Notably, we also had overfitting in the SFT setting for both datasets. As shown in Figure 4, the training reward steadily decreases over all 10 epochs, while the best validation reward is at epoch 5 in both settings. This suggests that our dataset was not rich enough. The SFT "extracted" all of the possible information from the dataset, and then RL could only overfit on top of that.

7 Conclusion

In conclusion, our project suggested that distillation via RL from larger, more expensive models can slightly improve performance in Lean premise retrieval (at least in reranking). The main bottleneck in our process was having a fully representative dataset. In domains where generating data is cheap, distillation via RL can potentially be quite effective. However, in our case and in general for generating high quality math data, it was quite challenging to get a large corpus of new proofs from Aristotle.

One future direction for this line of work would be to do some type of hierarchical RL. It was shown in our work that the distillation approach via RL showed some improvement in the specific subdomain of measure theory. Given a general query, we would have one high level policy determine which subject area the query is in. Then, we could train a number of smaller models on specific subareas of Mathlib that correspond to these subject areas. We could further boost their performance with rewards from Aristotle. This could potentially be helpful, since the way theorem similarity is represented semantically may be vastly different across different mathematical domains.

8 Team Contributions

- **Alexander López**: Worked on the dataset creation (both extraction from Mathlib's source files, natural language translation with DeepSeek), and implemented the embeddings RL loop.
- **Fred Rajasekaran**: Implemented the reranker system and helped with data generation.
- **Kevin Rizk**: Ran experiments and developed the system for returning the top-k search items using the Mathlib embeddings, and implemented the Aristotle data generation pipeline.

Changes from Proposal There were no major changes from the proposal. The team members finished the remaining tasks that were described in the proposal.

9 AI Usage

Coding agents were used with setting up Modal and WandB as well as for some aspects of data processing and most of data collection / translation. ChatGPT was used to proofread and find typos in the final report.

References

Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathïs Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, Martin Michelsen, Riley Patterson, Eric Rodriguez, Laura Scharff, Vikram Shanker, Vladmir Sicca, Hari Sowrirajan, Aidan Swope, Matyas Tamas, Vlad Tenev, Jonathan Thomm, Harold Williams, and Lawrence Wu. 2025. Aristotle: IMO-level Automated Theorem Proving. arXiv:2510.01346 [cs.AI] https://arxiv.org/abs/2510.01346

Leni Aniva, Chuyue Sun, Brando Miranda, Clark Barrett, and Sanmi Koyejo. 2024. Pantograph: A Machine-to-Machine Interaction Interface for Advanced Theorem Proving, High Level Reasoning, and Data Extraction in Lean 4. arXiv:2410.16429 [cs.LO] https://arxiv.org/abs/2410.16429

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The Lean Theorem Prover (System Description). In *Automated Deduction - CADE-25*, Amy P. Felty and Aart Middeldorp (Eds.). Springer International Publishing, Cham, 378–388.

Guoxiong Gao, Zeming Sun, Jiedong Jiang, Yutong Wang, Jingda Xu, Peihao Wu, Bryan Dai, and Bin Dong. 2026. LeanSearch v2: Global Premise Retrieval for Lean 4 Theorem Proving. arXiv:2605.13137 [cs.IR] https://arxiv.org/abs/2605.13137

The mathlib Community. 2020. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs* (New Orleans, LA, USA) *(CPP 2020)*. Association for Computing Machinery, New York, NY, USA, 367–381. doi:10.1145/3372885.3373824

Job Petrovčič, David Eliecer Narvaez Denis, and Ljupčo Todorovski. 2025. Combining Textual and Structural Information for Premise Selection in Lean. arXiv:2510.23637 [cs.LG] https://arxiv.org/abs/2510.23637

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. 2025. REAL-Prover: Retrieval Augmented Lean Prover for Mathematical Reasoning. arXiv:2505.20613 [cs.CL] https://arxiv.org/abs/2505.20613

Yicheng Tao, Haotian Liu, Shanwen Wang, and Hongteng Xu. 2025. Learning an Effective Premise Retrieval Model for Efficient Mathematical Formalization. arXiv:2501.13959 [cs.CL] https://arxiv.org/abs/2501.13959

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems* 36 (2023), 21573–21612.