Combining Textual and Structural Information for Premise Selection in Lean

Job Petrovčič
Faculty of Mathematics and Physics, University of Ljubljana
Jadranska 21, 1000 Ljubljana, Slovenia
jp10210@student.uni-lj.si

David Narváez
Faculty of Mathematics and Physics, University of Ljubljana
Jadranska 21, 1000 Ljubljana, Slovenia
david.narvaez@fmf.uni-lj.si

Ljupčo Todorovski
Faculty of Mathematics and Physics, University of Ljubljana
Jadranska 21, 1000 Ljubljana, Slovenia
Department of Knowledge Technologies, Jožef Stefan Institute
Institute for Mathematics, Physics and Mechanics
ljupco.todorovski@fmf.uni-lj.si

Abstract

Premise selection is a key bottleneck for scaling theorem proving in large formal libraries. Yet existing language-based methods often treat premises in isolation, ignoring the web of dependencies that connects them. We present a graph-augmented approach that combines dense text embeddings of Lean formalizations with graph neural networks over a heterogeneous dependency graph capturing both state–premise and premise–premise relations. On the LeanDojo Benchmark, our method outperforms the ReProver language-based baseline by over 25% across standard retrieval metrics. These results suggest that relational information is beneficial for premise selection.

1 Introduction and related work

Recent advances in artificial intelligence, particularly large language models (LLMs), have demonstrated increasing efficacy in formal mathematics and interactive theorem proving [Polu and Sutskever, 2020, Jiang et al., 2021, Xin et al., 2024]. A central task in this domain is *premise selection*: retrieving relevant theorems and definitions from extensive libraries to guide the proofs of new theorems. Effective premise selection underpins many automated reasoning tools, such as Sledgehammer [Böhme and Nipkow, 2010] and modern AI-driven provers for Lean [Song et al., 2023].

LLM-based approaches such as ReProver [Yang et al., 2023] use dual-encoder models to map proof states and premises into a shared vector space, retrieving relevant premises via dense ByT5 embeddings. However, they ignore the structural relationships in formal libraries: the references and dependencies among axioms, theorems, lemmas, and definitions (entries), which offer strong prior knowledge for guiding proofs. Attempts to exploit this structure include [Bauer et al., 2023] and [Ferreira and Freitas, 2020], but the former neglects Lean 4's state–tactic framework, while the latter tackles premise selection only in natural text.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MATH-AI.

We propose a methodology to integrate structural information into language-based premise retrieval for Lean 4. Our contributions are the following:

- We extend LeanDojo's dataset extraction to construct a heterogeneous dependency graph from Lean source files that allows for graph-enhanced premise selection in Lean's tactic mode.
- We design a simple relational graph neural network (RGCN) architecture to propagate graph structural information, producing graph-aware premise and proof-state representations. We demonstrate improved performance over the ReProver baseline on the LeanDojo benchmark, highlighting the benefit of incorporating dependency structure.

2 State-premise dependency graph for Lean

2.1 Augmented data extraction from the Mathlib Library

In tactic-based interactive theorem proving, the *proof state* at any point inside a proof represents the statement that currently remains to be proven (the *goal*), together with its local hypotheses. In Lean 4 one enters the tactic environment with the `by` keyword. When a user applies a tactic, this state is updated to reflect the resulting subgoals. Figures 2 and 3 show the proof states as displayed by Lean 4 for the example theorem in Figure 1. The first proof state in Figure 2 is the initial state after the `by` keyword. After applying the tactic `rw [← not_or]` to this state, we obtain the second proof state depicted in Figure 3.

```
theorem premise_example -- (a) premise node
  (p q : Prop) (h : ¬ (p ∨ q)) -- (b) signature hypotheses
  : ¬ p ∧ ¬ q := by -- (c) signature goal
  rw [← not_or] -- (d) proof -step dependency
  exact h -- (d)
```

Listing 1: Lean theorem using listings

Figure 1: A hypothetical Lean 4 theorem illustrating the extraction of the graph components. (a) The theorem `premise_example` becomes a premise node. (b) Signature hypotheses (with edges to premises `Or, Not`) and (c) the goal (with edges to premises `And, Not`) define signature dependency edges. (d) The tactic application creates a proof-dependency edge to premise `not_or`. The proof states (e.g., after the `by` keyword, Figure 2) become graph nodes linked to premises in their local hypotheses and goals.

```
p q : Prop -- (e)
h : ¬ (p ∨ q)  -- (e)
⊢ ¬ p ∧ ¬ q  -- (f)
```

Figure 2: Initial proof state immediately after `by`. *State nodes* are created for proof states. The initial state after `by` (in Figure 2) is linked to its local hypotheses (`[Or, Not]`) and goal premises (`[And, Not]`).

```
p q : Prop -- (g)
h : ¬ (p ∨ q) -- (g)
⊢ ¬ (p ∨ q) -- (h)
```

Figure 3: The updated proof state obtained after applying the tactic `rw [← not_or]`.

Base dataset: proof states and premises In the LeanDojo machine learning framework [Yang et al., 2023], premise selection is formulated as the following task. For each proof state s, given its textual representation x_s, we aim to identify the list of relevant premises y_s from the (Mathlib [The mathlib Community, 2020]) library that will be used in the next tactic. For example, for the proof state in Figure 2, the tactic uses the premise `not_or`. Although only one premise appears in this example, in general, a tactic application may use multiple premises or none. Let S denote the set of proof states, and X_S the text representations for each $s \in S$.

Let P be the set of all potential premises available in the library. Each premise $p \in P$ has a library definition in Lean code, and we denote by X_P the code (text) representations of the premises in

P. For example, in Figure 1, the entire block defines the premise `premise_example`. The dataset in [Yang et al., 2023] can therefore be summarized as the tuple (X_P, X_S, Y_S), where Y_S denotes the set of all lists of relevant premises y_s.

Graph structure Definitions of premises, as well as proof states, may reference previously defined premises. The graphical user interface for Lean 4 allows a user to navigate to a previously defined premise by clicking on its symbol in the formalization code of a premise or a proof state. We treat the underlying navigation links as references in our graph.

By merging $V = P \cup S$ as the node set, $X = X_S \cup X_P$ as the textual representations of the corresponding Lean formalizations, and references $E \subseteq V \times \mathcal{R} \times V$ as the edge set, we can summarize a Lean library as a text-attributed *directed* graph $G = (V, X, E)$. Here $\mathcal{R}$ is the set of relation types described in the next paragraph. We extend LeanDojo's open-source extraction framework [Yang et al., 2023] to query the proof assistant for this additional information and extract this full dataset, i.e., the tuple (G, Y_S). The dataset (for Mathlib) thus now includes (X_P, X_S, Y_S) from [Yang et al., 2023], but adds the additional premise-premise and premise-state edges, forming a directed graph. The modified extraction code and the learning pipeline used is available at `https://github.com/JobPetrovcic/GNNReProver/tree/lighweight`.

Relation types References can be categorized into different types depending on where premises appear in a definition or proof state. Each definition consists of:

1. The name ((a) in the example in Figure 1).
2. The signature—which itself splits into local hypotheses (b) and the goal (c).
3. The proof (d), required for theorems, lemmas, and definitions, but not for axioms.

A relation is thus assigned a type based on the positions of its occurrence, and we denote the set of these relation types by $\mathcal{R}$.

Proof states follow a similar structure: they consist of local hypotheses ((e) and (g) in the examples in Figure 2 and 3) and goals ((f) and (h)). The proof component is represented by the next tactic applied to this state. The premises the tactic uses are the lists y_s introduced earlier.

File and import graph To remain consistent with the framework of [Yang et al., 2023], we also use the separate directed graph of imports between the files and a map between entries, states, and the files where they were defined. We do not use this information during training, and it is not employed by the model. We leave the utilization of this information for future work. However, during evaluation, this graph is used to restrict the premise selection only to premises that would have been available to the model at that point in the file: premises from *imports*, and premises defined *before* the current entry. For complete details, see [Yang et al., 2023].

2.2 Dataset statistics

Table 3 in the appendix summarizes the graph statistics for the augmented LeanDojo benchmark extracted from the Mathlib library commit `29dcec074de168ac2bf835a77ef68bbe069194c5`, the one used in the official repository of [Yang et al., 2023]. This allows us to directly compare our results with the results of their approach.

3 Methodology

3.1 GNN-augmented premise retrieval

We can now formulate premise selection as learning a scoring function $f \colon S \times P \to \mathbb{R}$ that measures the relevance of premise $p \in P$ for proof state $s \in S$. To this end, we apply GNN-refined embeddings as follows.

Step 1: initial text embeddings. The textual representations of premises x_p and states x_s are initially embedded using ReProver's ByT5 dual-encoder [Yang et al., 2023], yielding initial node feature vectors $\mathbf{h}_p^{(0)}$ and $\mathbf{h}_s^{(0)}$, respectively.

Step 2: GNN-based refinement. We employ a *Relational Graph Convolutional Network* (RGCN) [Schlichtkrull et al., 2018] to propagate information over the heterogeneous directed graph. Each premise node's embedding $\mathbf{h}_p$ is updated iteratively using

$$\mathbf{h}_p^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_p^{(l)} + \sum_{(u,r,p)\in E} \frac{1}{\mathcal{N}_r(p)}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)}\right)$$

over L layers, where $\mathcal{N}_r(p) = |\{u|(u,r,p) \in E, u \in P, p \in P\}|$, σ is an activation function, and $\mathbf{W}_r^{(l)}$ are trainable RGCN weight matrices.

Step 3: GNN-refined state encoding. At retrieval, the proof state is treated as a temporary query node s connected to its premises. Using the same architecture with different weights, the same one-step message passing produces the embeddings of the states $\mathbf{h}_s^{(l+1)'}$ by aggregating embeddings $\{\mathbf{h}_p^{(l)'}\}$ of premises referenced by the state. Note that the edge directions prevent information flow from states to dependencies or future premises to prior dependencies.

3.2 Training objective

We use the InfoNCE [van den Oord et al., 2018, Rusak et al., 2025] loss, which for a given list y_s of valid premises, calculates as:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{\sum_{s\in S}|y_s|}\sum_{s\in S}\sum_{p\in y_s}\ln\frac{e^{\mathrm{sim}(\mathbf{h}_s^{(L)'},\mathbf{h}_p^{(L)})/\tau}}{\sum_{k\in P} e^{\mathrm{sim}(\mathbf{h}_s^{(L)'},\mathbf{h}_k^{(L)})/\tau}}, \quad \mathrm{sim}(\mathbf{u},\mathbf{v}) = \frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|},$$

where the hyperparameter τ is a scalar temperature. Note that by minimizing this loss, we train a model of the scoring function $f(s,p) = \mathrm{sim}(\mathbf{h}_s^{(L)'}, \mathbf{h}_p^{(L)})$. The InfoNCE loss contrasts positive and negative premises for each state, summing probabilities over multiple valid premises. The GNN is trained transductively on the full premise graph, excluding all edges from proof relations (the third item in the enumeration of relation types in Section 2.1) to prevent trivial memorization and promote learning of general structural patterns.

3.3 Other optimizations

We use an ensemble of six independently-trained models (N=6), averaging their outputs to form the final prediction. This approach mitigates initialization sensitivity, which is a known issue in GNNs [Li et al., 2023]. In addition, we apply exponential model averaging (EMA) to further improve performance and generalization [Morales-Brotons et al., 2024]. Finally, compared to LLMs, GNNs use relatively less memory, removing the need to sample negative examples. We thus simply use all other premises in the library as negatives.

4 Experiments and results

4.1 Experimental setup

For our experiments, we use the LeanDojo Mathlib benchmark dataset with graph augmentation. We adopt the same train/validation/test split as [Yang et al., 2023] (the "random" split) and evaluate with the same metrics: Recall@1, Recall@10, and Mean Reciprocal Rank (MRR). Our baseline is the ReProver retriever with a ByT5-small encoder. We tune our model's hyperparameters using Optuna on the validation set (see Appendix B). The number of GNN layers used is $L = 2$.

4.2 Results

Results reported in Table 1 show that our GNN-augmented retriever outperforms the baseline across all metrics. Note that the baseline results differ from those reported in [Yang et al., 2023]. On GitHub[1], the authors explain that this is potentially due to improvements made to the ReProver system from the time of their publication.

[1] `https://github.com/lean-dojo/ReProver/discussions/51`

Model	R@1	Δ	R@10	Δ	MRR	Δ
ReProver (Baseline)	13.42%	–	39.60%	–	0.3283	–
GNN-augmented retriever (Ours)	17.98%	+33.98%	50.04%	+26.36%	0.4095	+24.73%
Ours + EMA	**18.31%**	**+36.44%**	**50.33%**	**+27.10%**	**0.4140**	**+26.10%**

Table 1: Retrieval performance on LeanDojo test set. Relative improvements Δ are w.r.t. ReProver (Baseline).

4.3 Ablations

To understand the contribution of different components in our model, we perform an ablation study on the augmented LeanDojo data set. Specifically, we examine the effect of excluding the graph between contexts and premises and the graph between premises and premises. Table 2 summarizes the results of the ablations. Note that due to computational cost, no ensembling was used in all cases for fair comparison and thus the results for No ablation in Table 2 do not match those in Table 1.

Model	**R@1**	**R@10**	**MRR**
No ablation	17.43%	48.52%	0.4010
No ablation + EMA	17.74%	48.70%	0.4048
Context graph ablation	17.30%	49.99%	0.4008
Context graph ablation + EMA	17.58%	50.13%	0.4057
Premise graph ablation	17.45%	49.30%	0.4008
Premise graph ablation + EMA	18.18%	49.98%	0.4096

Table 2: Ablation study on the LeanDojo test set.

After performing ablations, we realized that removing parts of the dependency graph even improves performance. This leads to the possibility that the utilization of the graph structure is not the main factor behind the superior performance, but rather that this is caused by the different choices of the loss function and sampling strategy. We leave this scrutinization for future work.

5 Conclusion

We introduced a graph-augmented language approach to premise selection in Lean. By extracting fine-grained syntactic dependencies and propagating structural information via a GNN, our method produces embeddings that outperform text-based baselines.

As the ablation section suggests, however, the superior performance might be attributed to the different training paradigm choices rather than the model's utilization of graph structure. Besides further analysis of the root cause of the improvement, future work includes exploring advanced GNN architectures, such as graph attention networks, to better incorporate structural information. Finally, the model will be evaluated on more realistic splits, such as the LeanDojo "novel" split or a split based on the creation time of premises.

6 Acknowledgments

The authors acknowledge the financial support of the Slovenian Research Agency via the Gravity project *AI for Science*, GC-0001 and via the research core funding No. P2-0103, the Air Force Office of Scientific Research under award number FA9550-21-1-0024, and the Renaissance Philanthropy via the project *Bridging AI, Proof Assistants, and Mathematical Data (BRIDGE)*. We are especially grateful to Boshko Koloski for the beneficial discussion related to the training of our model.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Andrej Bauer, Matej Petković, and Ljupčo Todorovski. MLFMF: Data sets for machine learning for mathematical formalization. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Sascha Böhme and Tobias Nipkow. Sledgehammer: Judgement day. In *International Joint Conference on Automated Reasoning*, pages 107–121. Springer, 2010.

Deborah Ferreira and André Freitas. Premise selection in natural language mathematical texts. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7365–7374, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.657. URL `https://aclanthology.org/2020.acl-main.657/`.

Albert Qiaochu Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. LISA: Language models of Isabelle proofs. In *Proceedings of the 6th Conference on Artificial Intelligence and Theorem Proving (AITP)*, pages 17.1–17.3, 2021. URL `https://aitp-conference.org/2021/abstract/paper_17.pdf`.

Jiahang Li, Yakun Song, Xiang Song, and David Wipf. On the initialization of graph neural networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19911–19931. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/li23y.html`.

Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=2M9CUnYnBA`.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *CoRR*, abs/2009.03393, 2020. URL `http://arxiv.org/abs/2009.03393`.

Evgenia Rusak, Patrik Reizinger, Attila Juhos, Oliver Bringmann, Roland S. Zimmermann, and Wieland Brendel. InfoNCE: Identifying the gap between theory and practice. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan, editors, *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 4159–4167. PMLR, 03–05 May 2025. URL `https://proceedings.mlr.press/v258/rusak25a.html`.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Towards large language models as copilots for theorem proving in lean. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*, 2023. URL `https://arxiv.org/abs/2404.12534`.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, pages 1–10, New Orleans, LA, USA, January 2020. ACM. ISBN 978-1-4503-7097-4. doi: 10.1145/3372885.3373824. URL `https://doi.org/10.1145/3372885.3373824`.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Huajian Xin, Daya Guo, Zhihong Shao, Z.Z. Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. Advancing theorem proving in LLMs through large-scale synthetic data. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. URL `https://arxiv.org/abs/2405.14333`.

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. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

A Dataset statistic for Mathlib 4

Table 3: Summary statistics of the augmented LeanDojo Mathlib 4 dependency graph, including node counts and edge types used for premise and state relations.

Node Statistic	**Value**	
Total Number of Nodes	440,487	
Premise Nodes	180,907	
State Nodes	259,580	
Edge Statistics	**Premise-to-Premise**	**Premise-to-State**
Signature local hypotheses edges	652,484 (36.9%)	2,670,304 (63.4%)
Signature goal edges	626,318 (35.4%)	1,539,899 (36.6%)
Proof-dependency edges	490,356 (27.7%)	/
Next tactic premise labels (Y_S)	/	379861
Total Edges	1,769,158	4,210,203

B Hyperparameters

B.1 Hyperparameter tuning

The hyperparameters for our GNN retrieval model were determined through a two-stage search using the Optuna framework [Akiba et al., 2019]. The first stage focused on architectural choices, such as GNN layer type, hidden dimensions, learning rate, and model structure (e.g., separate vs. shared GNNs for premises and contexts). The objective was to maximize the Recall@10 metric on the *training set for the first batch*.

As the best model was prone to overfitting, we introduced a second stage, fixing the best architecture from the previous stage, and performed a fine-grained search for optimal regularization parameters. This included tuning the node feature dropout rate, the edge dropout rate, and the L2 weight decay for the optimizer. The objective in this case was to maximize the Recall@10 metric on the *validation* set.

B.2 Model and training configuration

Table 4 lists the final configuration of our best-performing model, obtained through a two-stage Optuna search described in the previous section. We train for 120 epochs and select the model with the best validation Recall@10 for evaluation on the test set.

C Resources

We ran all experiments on a cluster with three NVIDIA A6000 GPUs (48 GB each), two Intel Xeon Silver 4410Y CPUs (24 cores, 48 threads), and 512 GB of RAM.

The initial hyperparameter tuning stage, with reduced training epochs, took about one day, and the subsequent stage about. 12 hours. Final training of the ensemble of six independently trained models took roughly one day, with exponential moving average (EMA) optimization adding negligible time.

Table 4: Final model and training configuration for the GNN-augmented retrieval model.

Category	**Parameter**	**Value / Notes**
GNN	number of layers	2
	hidden size	1024
	activation	ReLU
	dropout	0.256
	edge dropout	0.142
	residual connections	Used
Loss	InfoNCE temperature	0.0138
Optimizer	learning rate	0.00499
	weight decay	2.359e-5
Training	batch size	1024
	gradient accumulation	2 batches
	epochs	120