Llama-PVS: A Lightweight decoder-only LLM for Proof Recommendation system

Nikson Bernardes Fernandes Ferreira

Universidade de Brasília - UnB

niksonber@gmail.com

Abstract

Interactive theorem provers (ITPs) are crucial for certifying the correct behavior of systems. ITPs require a higher level of expertise and present a steep learning curve, even for individuals with a strong background in both practical and theoretical aspects. In recent years, considerable research has been conducted to promote the use of machine learning techniques for suggesting proof commands and providing hints to ITP users. Such research builds recommendation models based on exhaustive training over translations of proof data history generated from a core of mechanized theorems. Subsequently, the ITP user can profit from the model recommendations by translating the status of a proof in progress to obtain feedback from the trained model after it is translated back to the proof language of the ITP. This double translation implies an extensive consulting preprocessing, consuming considerable resources. The present work skips the translation process and trains an autoregressive language model receiving the proof state directly. The approach evaluates the capability of these models to "understand" and extract patterns from raw proof, achieving competitive answer quality and performance.

Introduction

Machine Learning (ML) has been advancing for decades. In particular, advances in parallel computational resources have allowed the development and scaling up of complex deep learning techniques, such as Large Language Models (LLMs). Such models have been successfully applied to a diverse range of applications, including translation, question-answering, and code generation, giving rise to attractive application opportunities in automated reasoning and assisted theorem proving [2].

This work focuses on the integration of ML techniques and interactive proof assistants to improve the grade of automation through the generation of proof recommendations. The integration of deep learning and theorem proving has the potential to advancing proof techniques, such as a neuro-symbolic therem proving, enable development of more efficient and user-frindly proof assistant that could help the advance of a diverse range of fields, such as number theory, criptography, and even, can be used for education proposes allowing the development of tools to teach formal methods.

Objectives

In summary, this work presents the following Objectives:

1. A methodology that integrates LLM into the proof assistant PVS to produce proof commands recommendations during the interactive proving exercise.
2. In addition to guiding users in the interactive, effective selection of adequate proof commands, the methodology also proposes proof-step parameters, such as correct identification for a universal quantifier or evidence for existential, and lemma usage suggestion selected from the core of available lemmas in the NASA PVS library.
3. The imodel integration with `VScode` using the extensions of `llama.cpp` and `PVS`.
4. The validation of the need for extensive preprocessing used in other works that integrate Machine Learning and Interactive Theorem provers.
5. Compare the present work that proposes an integration of LLMs and PVS [7]

Background

LLM

Language Models (LMs) model statistical patterns in human-like languages. Fundamentally, they aim to predict the probability of the occurrence of a given token (word or sub-word) in a given context (text). Most modern LMs leverage a deep neural network architecture called *Transformers*. The Transformer architecture introduced in [6] is based on the self-attention concept used to update the meaning of tokens based on the meaning of context tokens. The self-attention mechanism is a major innovation in Transformer architecture. It enables learning the impact of tokens on other ones' meaning in parallel, allowing scaling performance on the so-called *Large Language Models* (LLMs), by extracting complex statistical patterns and language nuances on unprecedented levels.

LLMs are usually trained in two stages: an extensive self-supervised training on a huge corpus and a supervised fine-tuning on smaller and task-specific datasets. In the first self-supervised training phase, the model learns complex token relations, and in the fine-tuning phase, it transfers the previous knowledge to learn specific tasks, such as question-answer, translation, coding, etc. Besides being used for text-to-text tasks, LLMs do not generate text themselves, but they only predict the probability of the next token in a sequence. An external algorithm integrates LLMs to select and concatenate text pieces, constructing text interactively.

Training or getting outputs from these LLMs demands exhaustive computational power. In this way, several works have been proposed aiming to reduce the resources required to adjust the model to intended behavior [1]. LoRa (LOw Rank AdAptation) allows fine-tuning a model, adjusting only a small set of added parameters corresponding fraction of the original weights [4]. The main idea of the approach is to fit a decomposed matrix into a low rank through matrix multiplication, which is added to the original weights that remain frozen. Aiming to reduce the computational power required for model inference, researchers proposed quantization techniques, such as GPTQ [3], reducing precision for representing certain weights.

ITPs: PVS

Interactive theorem provers (ITPs) are sophisticated software systems designed to assist in the formal verification of mathematical theorems, hardware designs, and critical software systems. Introduced by SRI International in 1996, the Prototype Verification System (PVS) is a complete formal specification and verification environment. Mainly composed of a strongly-typed specification language supporting higher-order logic and an interactive theorem prover with a robust automated deduction engine [5]. Unlike other ITPs, such as Isabel, PVS does not have high-level of proof automation.

Proposed Integration: Llama-PVS

The Figure 1 shows the workflow proposed in the present work for proof-step prediction. The approach can be separated into two phases delimited by the boxes: Training (occurs offline) and Inference (occurs online, i.e., during proof). The first step of the training phase uses consolidated proof libraries jointly with the ITP Tool PVS Traces to execute consolidated proofs and collect proof goals and steps (proof traces). The data comprises all intermediary proof goals and steps of formulas from the selected library. This data is used to fine-tune a **pre-trained LLM** model using the framework `transformers`. The output of this phase is a **Trained LLM** that learned the structure patterns of PVS proofs. The Inference Phase is responsible for obtaining an actual proof-step recommendation from the system. An integrated development environment (IDE) is used to integrate the **ITP PVS interface** and model predictions. In this work, we propose the use of the widely used IDE Visual Studio Code (VSCode) and its PVS extension as the PVS Interface. The current proof goal is submitted to the model through an intermediary server to obtain a list of proof-step recommendations according to model certainty. Here, the inference framework **llama.cpp** is used for low-level interaction with the **Trained LLM**.

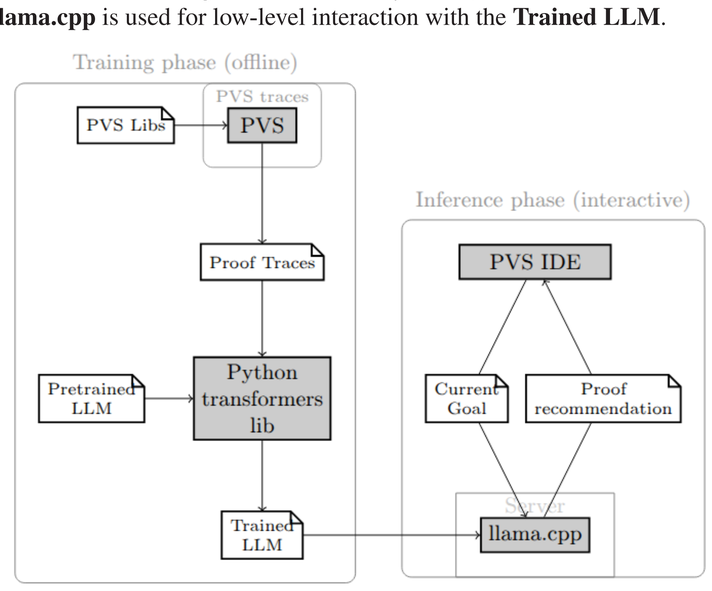

Figure 1: Workflow for training and integrating LLM for proof step prediction in an interactive theorem prover.

Additionally, Figure 2 outlines the workflow to train and develop a system for lemma recommendations. The workflow is also separated into the training and inference phases. As in the previous workflow, we first collect proof traces from selected libraries, but all lemmas used in the proofs in the selected library are identified, and their respective specifications (in their corresponding libraries) are collected and indexed. Then, all pairs of lemmas with their respective specifications are associated with the proof goals where the proof step invoked them. In the second training, the model learns to transform the text into a semantic vectorial space where proof sequent and lemma definition that can be used are approximated as the projection of its vectors. During interference, all lemmas are previously represented and stored in the vector space. The PVS interface provides the current sequence is also vectorized, then using cosine similarity between vectors, the system proposes the most similar ones to user's usage.

Figure 2: Workflow for training and integrating LLM for Lemma prediction in an interactive theorem prover.

Results

Following the proposal in the previous section, the models were trained using LoRA. The command prediction was evaluated in terms of accuracy, i.e., comparing the rate of predictions that match with the expected command, that is, the one used in the actual proof stored. Accuracy@k, means that the correct command is in the top-k commands suggested. The present work was compared to [7], a approach that uses a extensive preprecessing in proof sequent, ware in the present work, we use the raw sequent as input.

Metric	CoProver	CoProver[1] (our data)	PVS-Llama
Accuracy@top 1	23.37	48.33	55.56
Accuracy@top 3	45.95	76.43	82.72
Accuracy@top 5	57.82	85.91	90.92
Accuracy@top 7	66.27	91.39	95.21
Accuracy@top 10	72.50	95.32	97.17

Table 1: Comparing CoProver and PVs-Llama command prediction results.

For lemma suggestion, the MRR (Mean Reciproval rank), Recall, and MAP were used to compare the approaches. For comparison, one limit the search to a specific library (sorting) and all ones.

Metric	CoProver	PVS-Llama Sorting (ours) (our data)	PVS-Llama All (ours)
MRR	0.51	0.75	0.63
Recall@top 3	-	0.61	0.52
Recall@top 5	-	0.88	0.68
Recall@top 7	-	0.95	0.80
Recall@top 10	-	0.99	0.84
MAP@100	-	0.75	0.63

Table 2: Comparing CoProver and PVs-Llama lemma prediction results.

Conclusion and Future Works

A distinguished feature of the methodology is that the training is performed over raw proof states, i.e., the PVS proof sequents, applying QLoRA (Quantized Low-Ranking Adaptation) to an LLM. And subsequently, the model iteratively generates proof recommendations based on the current raw proof states. Besides of removing the pre-processing, we are able to get stronger results.

Still pending experiments with a bigger set of libraries and perform repetitions of experiments.

References

[1] Shengnan An, Yifei Li, Zeqi Lin, Qian Liu, Bei Chen, Qiang Fu, Weizhu Chen, Nanning Zheng, and Jian-Guang Lou. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv preprint arXiv:2203.03131*, 2022.

[2] Lasse Blaauwbroek, David M. Cerna, Thibault Gauthier, Jan Jakubuv, Cezary Kaliszyk, Martin Suda, and Josef Urban. Learning guided automated reasoning: A brief survey. In Venanzio Capretta, Robbert Krebbers, and Freek Wiedijk, editors, *Logics and Type Systems in Theory and Practice - Essays Dedicated to Herman Geuvers on The Occasion of His 60th Birthday*, volume 14560 of *Lecture Notes in Computer Science*, pages 54–83. Springer, 2024.

[3] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[5] S. Owre, J. Rushby, and N. Shankar. PVS: A prototype verification system. In *Proceedings of the 11th International Conference on Automated Deduction, CADE*, pages 748–752. Springer, 1992.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[7] Eric Yeh, Briland Hitaj, Sam Owre, Maena Quemener, and Natarajan Shankar. CoProver: A Recommender System for Proof Construction. In Catherine Dubois and Manfred Kerber, editors, *Intelligent Computer Mathematics - 16th International Conference, CICM 2023, Cambridge, UK, September 5-8, 2023, Proceedings*, volume 14101 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2023.

Acknoledgments

This work was founded by CNPq.