Autoformalization with Backtranslation: Training an Automated Mathematician

Stanford CS224N Custom Project

Jakob Nordhagen
Department of Computer Science
Stanford University
jakobn@stanford.edu

Abstract

Recently, large language models (LLMs) have recorded impressive performance and versatility on neural machine translation tasks. As such, LLMs show significant potential for autoformalization, the task of translating proofs from informal mathematical language (IL) into formal mathematical language (FL). A success in this task would have significant and far-reaching implications, potentially allowing for the formalization and codification of all known mathematical textbooks, as well as establishing an avenue for automatic compute-driven advancement of the world's mathematical knowledge. The main problem is that vanilla LLMs currently lack robust quantitative reasoning capabilities, causing them to perform poorly on complex math tasks such as autoformalization and the downstream task of formal theorem proving. In this paper, we explore several approaches to teaching LLMs to reason mathematically by fine-tuning them for autoformalization. Specifically, we test the extent to which backtranslation (an unsupervised training method for automatically generating artificial labeled examples) is effective as a way to mitigate the problem of lacking labeled data which has hindered advancement in autoformalization. We also investigate the strategy of translating proofs one line at a time, allowing for a more granular learning process. We find that backtranslation leads to a nontrivial improvement in an LLM's performance on autoformalization, as measured on the ProofNet benchmark (Azerbayev et al., 2023a). Our methods yield positive results when used to fine-tune GPT-2 for autoformalization, showing promise for automated theorem proving and beyond.

1 Key Information to include

- Mentor: Nelson Liu
- External Collaborators: Michael Souliman, Willy Chan, Brando Miranda (advisor)
- Sharing project: CS197 (CS Research)

2 Introduction

Neural machine translation (NMT) has been an area of active research since the early days of machine learning and neural network architecture development. Autoformalization can be interpreted as a special case of NMT, where the task is to translate a theorem (or any other mathematical statement) from informal language, denoted IL, to formal language, denoted FL. Here informal language is in the form of text and LaTeX, and formal language is specified as Lean code. Lean4, the latest version proposed by de Moura and Ullrich (2021), is an open-source theorem prover and programming language designed to encode mathematics for the purpose of tasks such as programmatic theorem proving.

2.1 LLMs & Autoformalization

Despite considerable advancements in recent years, standard LLMs still struggle with quantitative reasoning and, by extension, with tasks like autoformalization and theorem proving. This is primarily due to the complex and sequential nature of mathematical reasoning, which often extends beyond the reach of the sort of first-order logic used in current automated theorem provers. These limitations highlight a gap in the ability of current models to understand and translate complex mathematical concepts and proofs from natural language to formal logic.

The widespread success of pretrained models inspired by the likes of GPT-3 (Brown et al., 2020) has led to an revolution in fine-tuning pretrained models on specific datasets to achieve (in many cases) superhuman performance on specialized tasks. However, fine-tuning for the task of autoformalization is quite challenging due to the lack of available data for training. Though large amounts of data exist in both formal language (through datasets such as Mathlib (DBL, 2019)) and informal language (through mathematical textbooks and other online resources), there is a distinct lack of parallel corpora with paired examples of IL and FL translations. This unique problem distinguishes autoformalization from other NMT tasks such as German-to-English, for example.

2.2 Data Augmentation and Backtranslation

Various techniques have been explored for data augmentation in cases where training data for supervised learning is scarce. Backtranslation, a method for generating artificial training data given a dataset in the target language, is one of the more prominent and proven methods for unsupervised NMT. Backtranslation proceeds as follows:

1. Use some pretrained "teacher" model to generate translations in the reverse direction, i.e. from the target language to the source language.
2. Assemble a paired dataset of these pairs.
3. Fine-tune or train some "student" model on the generated dataset, sidestepping the problem of lack of data.

Backtranslation is the primary method we explored for generating training data, allowing us to fine-tune a model in a semi-unsupervised fashion.

2.3 Interactive Theorem Proving & Autoformalization

Additionally, the use of Interactive Theorem Provers (ITPs) in extracting datasets of intermediate proof steps presents a novel opportunity for the task of autoformalization. By informalizing intermediate statements in a proof sequentially, we can generate a dataset that considers the individual steps undertaken by ITPs. This data can then be used to train an LLM, potentially replicating in some sense the sequential reasoning performed by ITPs. This would be a novel method in the field of autoformalization.

Finally, to assess the effectiveness of our autoformalizer and theorem prover, we evaluate our model on ProofNet's autoformalization benchmark (Azerbayev et al., 2023a). Being specially designed for the purpose, this benchmark is critical for us to assess our approach's ability to teach a model to translate natural language into formal language proofs accurately and efficiently. The success of our model on these benchmarks would demonstrate a significant advancement in the field of automated theorem proving.

2.4 Significance

The implications of successful research on autoformalization are significant and far-reaching. By developing a more effective autoformalizer using our dataset methods, we can help pave the way for a formal theorem prover capable of understanding theorems in natural language and converting them to formal, programmable language. Such a development could revolutionize the field of automated theorem proving, making it more accessible and effective in handling a broader range of mathematical problems. Furthermore, this research contributes to the broader understanding of how LLMs can be fine-tuned for specific, complex tasks like theorem proving, opening up new possibilities for their application in various scientific and mathematical domains.

Ultimately, our approach to enhancing LLMs for autoformalization and formal theorem proving represents a shift in the methodology of automated theorem proving. By focusing on individual proof lines and using backtranslation to generate a dataset that pairs natural language proofs with formal statements, we aim to bridge the gap between natural language understanding and formal logic reasoning. This research not only advances the field of automated theorem proving but also contributes to the broader goal of creating models capable of mathematical reasoning. Such a development could expand the field of mathematics in a way that has never been seen before.

3 Related Work

The idea of solving theorem-proving problems—and, more generally, quantitative reasoning problems—with neural networks has been explored with interest during the last decade, fueled in no small part by the revolution of the transformer architecture (Vaswani et al., 2017) and the subsequent explosion of LLMs. While LLMs have performed well on various natural language understanding tasks as evaluated by Hendrycks et al. (2020), they have continued to struggle with tasks that require deeper quantitative reasoning (Lewkowycz et al., 2022). Moreover, while strides have been made in using deep learning for symbolic mathematics (Lample and Charton, 2020), propositional logic (Hahn et al., 2021), and differential systems (Charton et al., 2021), the field of autoformalization and theorem proving has generally not seen proportional advances.

Autoformalization saw its first significant experiments in 2018 with the use of long short term memory networks (LSTMs) to generate statements in Mizar (Wang et al., 2018). Most recently, Wu et al. (2022) laid important groundwork for autoformalization with LLMs, showing promising results using naive few-shot learning on the autoformalization task, in their case translating English to Isabelle code.

In the realm of NMT, numerous methods have been explored for mitigating the issue of limited training data. For instance, Han et al. (2021) demonstrated the effectiveness of backtranslation for French-to-English translation, using the unsupervised method to new train state-of-the-art NMT models.

Lastly, the development of benchmarks for performance on reasoning tasks has considerably accelerated development in the area of teaching LLMs to reason more effectively. Rigorous general benchmarks for natural language understanding have been continuously used to stress-test the latest models. For instance, the Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2020) and General Language Understanding Evaluation (GLUE) (Wang et al., 2019) benchmarks are widely cited when evaluating and understanding the reasoning capabilities of LLMs. Recently, more specialized benchmarks have emerged that can evaluate performance specifically on mathematical tasks, including the MiniF2F (Zheng et al., 2021) and ProofNet (Azerbayev et al., 2023a) benchmarks. Developed from a comprehensive suite of Olympiad-level mathematics problems, these benchmarks are indispensable to the development of informal theorem proving models and indeed any model that performs autoformalization. We use both in our training and evaluation methods.

4 Approach

4.1 Neural Methods for Informalization

We set out to train an LLM on the formalization direction of the translation task, i.e. translating informal language (IL) to formal language (FL) in the form of Lean mathematical code. Given the very limited amount of labeled data or pairs of matching (IL, FL) examples to use as training data, we use few-shot amplification and backtranslation to fine-tune a model, loosely following the method described by Han et al. (2021).

Our backtranslation process proceeds as follows: we use a large pretrained model such as GPT-4 (OpenAI, 2023) as the "teacher" model, feeding it a dataset of FL examples and using it to create matching (artifical) IL examples. This is the informalization direction of the translation task, going from FL to IL. Here we leverage the fact that LLMs are excellent at in-context learning (DBL, 2019). We use a few-shot prompt that prefixes the informalization prompt with six labeled examples of FL-to-IL translations. We use this few-shot amplification method to create a large labeled dataset for training.

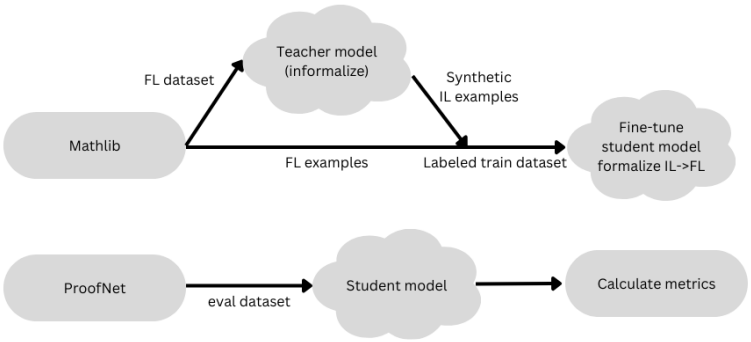

Figure 1: Diagram showing the flow of data in our backtranslation training method.

We then use this synthetic few-shot-generated training data to perform backtranslation, wherein a second pretrained large language model, denoted the "student" model, translates the synthetic IL examples back to FL. In this way, we fine-tune the student model on many matched pairs of informal and formal language; the hypothesis is that this teaches the model to formalize informal statements more accurately. Here, we used HuggingFace's Trainer API as well as the pretrained GPT-2 tokenizer and model (Wolf et al., 2020).

Finally, we evaluate our trained model on ProofNet's test dataset, calculating the cross-entropy loss. Figure 1 shows a visual representation of the training and evaluation pipelines.

4.1.1 Entire Proofs

Our first method followed the methods used by Azerbayev et al. (2023a). Concretely, we informalize on the entire-theorem granularity, including the entire proof as context for the prompt, and informalize the theorem statement (translate to informal language). For this method, we used our 6-shot prompt to generate a dataset of synthetic training pairs for backtranslation.

4.1.2 Individual Lines

Our second method implements the idea of considering proofs line-by-line to model better sequential reasoning. Here, we informalize only the theorem statement along with each individual tactic within a proof. By informalizing tactics one by one, we intended to test if this granular method could semantically represent, to an LLM, what was happening in each tactic and allow it to understand the proof better as a whole. For these informalizations, we used the LeanDojo dataset (Yang et al., 2023), which we parsed to extract the sequence of tactics taken along with the state of the proof before the tactic was applied and after the tactic was applied. We used a zero-shot prompt including tuples of (stateBefore, tactic, stateAfter) to generate a synthetic dataset of individually translated tactics using GPT-4.

4.2 On-the-fly Backtranslation

Finally, the most experimental concept we explored for the purpose of data augmentation was a form of backtranslation that we implemented within a custom training loop. This approach had no concept of separate teacher and student models; instead, we attempted to train the same model on both directions of the translation task at once, combining the loss from both and generating its own training data within each step. Concretely, during each training step:

1. Translate a batch of FL examples to IL.
2. Translate the synthetic IL examples back to FL, essentially using the (synthetic IL, FL) pair as training data.
3. Compute the loss of the generated FL translation compared to the original FL.
4. Backpropagate and update model weights.

The primary advantage of this method was its self-sufficient nature, allowing us to sidestep the problem of limited labeled data without incurring large API costs. However, the method suffered from the relatively small model used in our experiments; as it turned out, GPT-2's synthetic informalizations were not accurate enough to produce meaningful results when used as on-the-fly training data; in fact, the loss actually increased while training. Due to a lack of the necessary computational resources, we were ultimately unable to validate this method with a larger model such as Llemma-7B Azerbayev et al. (2023b) or Google's Gemma-7B model. We hypothesize that using a larger and more capable pretrained baseline could have produced promising results. We omitted this method, focusing instead on using large teacher models (GPT-4 and others) to generate synthetic training data for backtranslation, finding unsurprisingly that the results were significantly better given our small model size.

5 Experiments

5.1 Data

In the process of developing a robust dataset for the goal of improving math reasoning in LLMs, we adopted a systematic approach to gather diverse datasets. This was paramount in creating a robust foundation for our subsequent analyses and model training processes. Below we summarize our three methods of data augmentation for autoformalization:

1. Parsing individual proof tactics from LeanDojo with regular expressions: this method was instrumental in obtaining a preliminary dataset that served as the basis for initial model training. Importantly, this method afforded us precise control over the selection of tactics and proof methodologies, enabling targeted improvements in specific areas of interest.
2. Acquisition of formal and informal datasets from the miniF2F dataset: We used as much labeled training data as we could find, and compared it with our other data generation methods. Specifically, we extracted paired datasets from the MiniF2F benchmark's data, providing a valuable source of high-quality informalizations. This additional source of data was vital for enhancing our model's proficiency in complex proofs. However, the limited volume of this human-verified data underscored the importance of integrating unsupervised data generation methods such as few-shot amplification.
3. Few-shot amplification with GPT-4 to informalize data extracted from LeanDojo: By presenting GPT-4 with both the miniF2F dataset and textual data from LeanDojo's documentation, followed by an example proof from LeanDojo, we facilitated the generation of nuanced informalizations. Importantly, these generations included state information, as the provided proofs from LeanDojo encompassed the state before and after a tactic was applied, along with the tactic itself. This method was able to enhance the dataset's diversity by incorporating state information into the informalizations. The incorporation of state information was instrumental in broadening the variety of our training data, enabling our model to better grasp the intricacies of proof tactics and their impacts.

We used data collected through these three methodologies to form four datasets, described in the Evaluation section. These datasets were compared for effectiveness in training our model, ensuring a broad spectrum of proof tactics and informal statements was represented. A detailed breakdown of the dataset composition, including the total number of tokens and examples from each collection method, is provided in Table 1.

Data Collection Method	Number of Tokens
MMA Train	10,916,097
GPT-4 MathLib4 (Full Proof)	71,550
GPT-4 MathLib4 (Individual Tactics)	1,754
MiniF2F Original	36,056
MiniF2F (Individual Tactics)	23,761

Table 1: Token distribution across dataset generation methods. For each dataset generation method - Regex-Parsed LeanDojo Proofs, miniF2F Dataset, and GPT-4 Informalized LeanDojo Data - we compile formal and corresponding generated informal data into pairings. The model is trained and evaluated on each individual dataset to assess method-specific performance, as well as on the aggregated dataset to evaluate combined performance. The table presents the number of tokens contributed by each method to the overall dataset, providing insights into the diversity and scale of training data available for model optimization.

5.2 Evaluation method

We quantified the results of our methods by evaluating GPT-2 (before and after fine-tuning) for accuracy on ProofNet's test dataset, as measured by cross-entropy loss. By reserving a subset of data exclusively for testing, we ensured that our models undergo a fair evaluation. In this way, we were able to test different dataset generation methods against each other in a precise manner. We also include results from an existing autoformalization dataset, denoted MMA, for comparison (Jiang et al., 2023).

5.3 Experimental details

The first experiment that we ran was to evaluate GPT-4's viability as a teacher model. We developed a few-shot informalization prompt using examples from a dataset porting the MiniF2F test benchmark (Zheng et al., 2021) to Lean4 code (uploaded by the Hoskinson Center at CMU, found here on HuggingFace: https://huggingface.co/datasets/hoskinson-center/minif2f-lean4). We evaluated GPT-4's informalizations by manual comparison to the ground-truth IL examples, finding that GPT-4 generated reasonable translations for 28 out of 30 examples.

We created 4 synthetic datasets in total using the methods outlined above:

1. Few-shot amplification with GPT-4 on mathlib proof statements, using the whole proof as context
2. Zero-shot line-by-line informalization with GPT-4 on mathlib proofs
3. Extracted translations from the MiniF2F benchmark, treated on a whole-proof granularity
4. Extracted translations from the MiniF2F benchmark, treated on a line-by-line granularity

We ran fine-tuning code with the goal of comparing the datasets' effectiveness. Here we used AdamW as the optimizer, with a learning rate of $5 \cdot 10^{-5}$, batch size of 4, and a weight decay coefficient of 0.01. Due to compute constraints, we used a relatively small student LLM (GPT-2).

5.4 Results

For our experiments, we minimized the evaluation loss and used the ProofNet test dataset as our evaluation set. All of our models used GPT-2 as the base model and fine-tuned for 3 epochs. We included the results of fine-tuning on the training and validation sets of MMA (Jiang et al., 2023).Table 2 notes the evaluation loss measured.

Data Collection Method	Evaluation Loss
GPT-2 Baseline	75.1839
GPT-4 MathLib4 (Individual Tactics)	5.1287
Regex-Parsed LeanDojo Proofs	4.4709
MMA Train	3.8791
MiniF2F Original	3.5476
GPT-4 MathLib4 (Full Proof)	3.1209
MiniF2F (Individual Tactics)	**1.9915**

Table 2: Evaluation loss after fine-tuning GPT-2 on each dataset for 3 epochs.

6 Analysis and Conclusion

From the results we obtained, it is clear that our few-shot amplification and backtranslation method improves significantly over the baseline on GPT-2. However, we can also see that human-validated data, as in the datasets we extracted from MiniF2F, have a more profound effect on performance when used for training even though GPT-4's synthetic informalizations pass the visual test. Our line-by-line informalization method showed positive results on our MiniF2F dataset, but a negative effect on our few-shot-amplified mathlib4 dataset (see Table 2). We conclude therefore that the line-by-line sequential methodology boosts performance on high-quality, validated training data, but confuses the model when used on synthetic training data.

Qualitatively, our final models did not produce excellent translations, upon visual inspection. This is likely because of our student model's relatively small number of parameters. However, our methods produced a significant decrease in evaluation loss on the ProofNet benchmark and showed comparable performance to the MMA dataset despite being a tiny fraction of the size (see Table 1), validating them as promising avenues for training LLMs for autoformalization.

In future work, our methods could be tested on larger models with significantly more parameters, such as Code-LlaMa, Llemma, or Mistral. Additionally, there is much work to be done on sensible metrics for evaluating accuracy on the informalization task, as there is currently no automatic and accepted way to evaluate correctness of Lean proof statement formalizations.

References

2019. The lean mathematical library. *CoRR*, abs/1910.09336.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023a. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023b. Llemma: An open language model for mathematics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Francois Charton, Amaury Hayat, and Guillaume Lample. 2021. Learning advanced mathematical computations from examples. In *International Conference on Learning Representations*.

Christopher Hahn, Frederik Schmitt, Jens U. Kreber, Markus Norman Rabe, and Bernd Finkbeiner. 2021. Teaching temporal logics to neural networks. In *International Conference on Learning Representations*.

Jesse Michael Han, Igor Babuschkin, Harrison Edwards, Arvind Neelakantan, Tao Xu, Stanislas Polu, Alex Ray, Pranav Shyam, Aditya Ramesh, Alec Radford, and Ilya Sutskever. 2021. Unsupervised neural machine translation with generative language models only.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *CoRR*, abs/2009.03300.

Albert Q. Jiang, Wenda Li, and Mateja Jamnik. 2023. Multilingual mathematical autoformalization.

Guillaume Lample and François Charton. 2020. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Leonardo Mendonça de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *CADE*.

OpenAI. 2023. Gpt-4 technical report.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. 2018. First experiments with neural translation of informal to formal mathematics. In *Intelligent Computer Mathematics*, pages 255–270, Cham. Springer International Publishing.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2021. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*.