Ada-Math : Math Based Program Synthesis

Moya Zhu
Data Science Institute
Columbia University
mz2886@columbia.edu

Zihao Zhang
Data Science Institute
Columbia University
zz2763@columbia.edu

Saisamrit Surbehera
Data Science Institute
Columbia University
ss6365@columbia.edu

Abstract

In this paper, We explore the application of transformer-based Language Models (LM) to automated theorem proving. We specifically use a GPT-3 based neural network fine-tuned on mathlib code to generate math sequences. Our LMs acts as an automated prover and proof assistant. We further combine couple of language models and re-ranker to select for the best results. Overall this improves the results from 0.322 to 0.3627 for out GPT-3 based ADA model and 0.4592 for our combined GPT-3 and GPT-NEO model. Our github code is on. [1]

1 Introduction

Language modelling has vastly improved traditional Natural Language Processing tasks. Large language models like BERT Rae et al. (2021) have demonstrated improvement across reading comprehension, humanities, ethics, STEM, and medicine categories. I In mathematics text related language models, work can be done on creating a large scale pre-trained model which is trained on mathematics formulas and their contexts in addition to general texts to improve the performance. Peng et al. (2021)

But deep learning does not achieve relatively big success in tasks requiring extensive planning and symbolic reasoning. And math is one such domain which involves reasoning about the sequence of symbols but also has a strong semantic relation with its context. To handle formal mathematics text, there are two main challenges: Infinite action space and no direct self-play setup. Polu et al. (2022) Infinite action space means during the proof search, the model has to choose not from a well-behaved finite set of actions, but an infinite and complex set of tactics. No direct self-play setup means a prover is not playing against an opponent but a set of statement to prove. Allowing us to leverage provers to solve our problem.

So our project aims to tackle both of these challenges by using different decoding methods for language generation from multiple Transformers. To help guide our transformers we use data from libraries that MathLib and Lean consist of strategies or tactics. A tactic is a proofterm which acts as a representation of a mathematical proof. These are commands, or instructions, that describe how to build such a proof. These tactics are similar to programs or functions. Given the rise of language models in zero-shot program synthesis, we can apply improvements to the Math-Domain.

Together all the language models combined with different decoding methods which include Beam search, Temperature sampling and Top K sampling give us variety of results. To limit our domain, and try on a smaller dataset, we use concepts from Information Retrieval to Re-rank our results.

Overall, we would like to contribute by introducing the following models:

- GPT-Ada-Math: A fine tuned Ada model from OpenAI on the TACTIC datset. These include topics from various areas of Mathematics ranging from topology to linear algebra.

[1] https://github.com/nineTailsKurama/mathProof

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

- GPT-NEO-Math: A fine tuned GPT-NEO model which is post-trained on Mathlib and then Fine tuned on TACTIC Dataset.
- GPT-NEO-ReRanker: A fine tuned GPT-NEO model For Sequence Classification for re ranking out documents.

Overall all these models help us to get an accuracy of 0.4592 after combining the models. Our Re-Ranker without training ranks the results at top 6 on average. Although BERT is known to not rerank well we implement part of Localized Contrastive Estimation (LCE) as discussed in Gao et al. (2021) as to help get better results with minimal training.

2 Previous work

2.1 Language Models

Information extraction from textual data has been a centralized research problem in the past decades. Language models are the core to tackle this problem. A language model is a probability distribution over sequences of words, which provides context to distinguish between words. Given an input sequence, a language model is able to predict the likelihood of an output sequence. The goal of language modeling is to learn such a probability distribution from a training corpus. Recent improvements in transformers and pre-trained language models have provided a new direction to the field of language modeling. Pre-trained language models can be fine tuned for various language modeling tasks. We specifically use it for synthesizing tactics.

2.2 Lean lexical structure

Lean is predominately a functional programming language that makes it easy to write correct and maintainable code and acts interactive theorem prover. Lean essentially has a specific Lexical Structure. These include special modifications for symbols, identifiers, strings, characters, numerals and field notations. We will be using the Lean Gym [2] wrapper for our initial trials to verify the results. We use the lean proofs generated to fine-tune our model.

2.3 Math-BERT

MathBERT Peng et al. (2021) is a model from the BASE BERT model, which pre-trains on a large mathematical corpus ranging from pre-kindergarten, high-school, to college graduate level mathematical content. Original BASE BERT is trained on general domain texts like articles and Wikipedia pages and hence is hard to reach a good performance on mathematical text. Therefore, a special BERT model trained on mathematical domain corpora is proved to be effective on math related works and that is MathBERT. MathBERT was proved to demonstrate out-performance of BASEBERT on several popular NLP tasks including prediction of knowledge component, etc. Guo et al. (2019). We first worked on this approach

2.4 GPT-3

Generative Pre-trained Transformer 3 (GPT-3) is an auto-regressive language model that base on Transformers models. GPT-3 was trained on a large amount of text data. The model can be used to generate text, or to understand and interpret natural language. It's also one of the largest neural networks available, with 175 billion parameters. For the experiment, we worked on ADA version of this project which has 2 Million parameters.

2.5 GPT-F

GPT-F is a language model that applied transformer language model to automated theorem proving and address the reasoning capabilities of the deep learning models. The model was built with Metamath library as a formal environment. And the architecture was designed with decoder-only transformers that similar to GPT-3. A similar model trained with similar kinds of data has an accuracy of 32.2% on the TACTIC Dataset.

[2]https://github.com/openai/lean-gym

3 Data

The initial dataset contains of tactics by Lean and MetaLib, the GPT-F had approximately 3million of such proof steps for about 38k theorems as there are different proof labels. We further split the dataset into a train set and two valid and test sets each containing approximately 1k proofs sampled randomly (90k proof steps each).

3.1 Descriptions and Statistics

Given the time bounded scope of the tasks, we only used the proof-step objective.A proof-step objective is a goal and a tactic, potentially generating new subgoals. We use for training is a conditional language modeling objective that is asked to generate the proof-step given a GOAL, which is directly applicable to proof searches. A goal in this case is a statement in the context of a proof search.

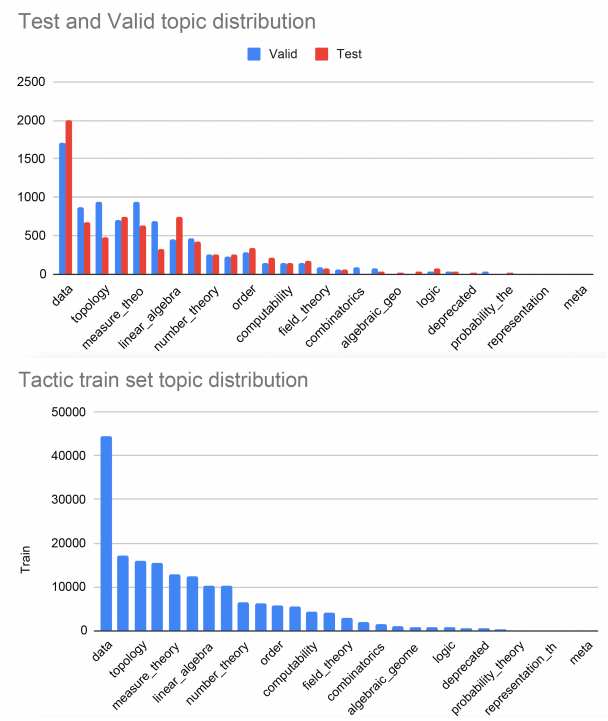

Figure 1: Data Districations based on the math topics

We summarized the distribution of mathlib tactics dataset topics 2. As the graph displayed, the general distribution for train, test and valid sets are very similar. However, the label heavily suffered from imbalance frequency (the top frequency for train set is over 40K and the least one have 1). The label imbalance leads to over-classify for the larger classes due to the increase probability.

4 Methods

Our approach has three different models. We combine all the models to form a pipeline where GPT3 Ada and GPT-NEO-MATH act are generators and GPT-NEO-ReRanker acts as a re-rankers. Re-Ranking is a unque process as we want new and diversified results for the Lean Prover.

4.1 GPT3 Ada Model

The GPT3 Ada model is a statistical machine learning model that is used to predict the probability of a particular outcome, based on a set of training data. The Training Objective for the Model for

training is to get a language modeling objective that is guesses the PROOFSTEP with a GOAL. With each goal as the sequence below.

<Start of Line> GOAL <GOAL> PROOFSTEP <PROOFSTEP> <End Of Line>

The fine-tuning of the models was done on OpenAI servers using the CLI tool. We used Mini-batch size of 2 as our model was really big.

4.1.1 Goal Expansion

For our goal expansion, we used temperature of 0.4 and max tokens 64 . We then de-duplicate them, and select for the valid tactics. This helps us in the proof search as we maintain a proof tree and a queue of open goals sorted by their cumulative log-prob. Each tree like the GPT-F paper when a successful tactic application generated new sub-goals. The proof tree and the proof search queue is then executed for each successful query.

4.2 GPT-NEO-Math Model

GPT-NEO-Math model designed using EleutherAI's replication of the GPT-3 architecture. We used the smallest version of 150 Million parameters as that took us the lowest time 3 days to train. The hidden size of the model was 768, it has 12 layers and 12 attentions heads. The vocab size and window size was 50259 and 256 respectively.

Since the GPT-NEO-Math model was the smallest model and context size , we had to remove the largest goals as they would not fit into memory and throw us error. We therefore limit our test set by about 600 large examples.

4.2.1 Goal Expansion

The Goal expansion for the GPT-NEO used different decoding methods. These decoding methods included Beam search, Temperature sampling and Top K sampling.

The reason behind Beam search is it reduces the risk of missing hidden high probability word sequences by keeping the most likely number of beams of hypotheses at each time step and eventually choosing the hypothesis that has the overall highest probability. This usually combines to give us the best results. We select 5 different beam search results for beam search model.

In Sampling means randomly picking the next word according to its conditional probability distribution. We specifically use Top-K sampling. In this process the K most likely next words are filtered and the probability mass is redistributed among only those K next words. This is the main reason we can see high variance in the type of output the GPT-NEO-Math Produces. Top-K sampling seem to produce more complete text than traditional greedy or beam search for our test set. Using all three gives us a really good variance of results.

4.3 GPT-NEO-ReRanker

One of the small problems with combining the language models results is the number of results grows exponentially. Since, we want the proover to get it right on the first try we use a Rerankers. We know that Rerankers fine-tuned from deep LM estimates candidate relevance quite accurately. The score based on rich contextualized matching signals.

We combine the results from the previous two model and then use the popular Reranker Library[3] with some modifications. The paper uses localized constrastive esimation (LCE) loss. In this the we select from a train set a postitive and negative examples of tactics. A negative and positive sample distribution by sampling from the target retriever top results. We combine this with a contrastive form loss which penalizes signals from confounding characteristics. We effectively want to get a wider array of tactics rather than focus on similar ones.

[3] `https://github.com/luyug/Reranker`

5 Results

In our process, we started by evaluating our model on MathBERT, based on our initial studies we have a loss of 3.15 and an training accuracy of 0.13 on our test set. Accuracy here refers the getting the tactic right. We had training loss as 3.164 and a loss of accuracy of 0.14 on the validation set. To better see our process, please take a look at Figure 1 and Figure 2.

Given the low accuracy of MathBERT, we selected GPT-3 and GPT-NEO to further fine-tune with the tactic proof dataset we selected. The training accuracy displayed below**??** shows the GPT-adaMath model we trained achieved the highest accuracy of 36.27%. GPT-3 training result is more refined and have more similar queries while GPT- neo on average have worse accuracy but the result have high variance. We come up with the intuition to combine both to leverage the variance, at at the same time apply re-ranker to select the first 6 queries to improve the frequency. The combined model end up getting the highest accuracy of 45.92% compare to the benchmark from GPT-f with accuracy 32.2%.

Models	Accuracy
GPT-f	32.2%
GPT-adaMath(3)	36.27%
GPT-NEO-Math	30.75%
combined(GPT-NEO-Math re-ranked)	45.92%

5.1 GPT-3

GPT-3 achieve the best accuracy result as showed in **??**. During the training we observe the test loss dropped in first 2.5 epoch and increase afterwards, which indicate model overfitting and we should do early stop at epoch 3. Unfortunately OpenAI doesn't have the option of early stopping. From the accuracy graph we can observe many big spikes. These are caused due to the mini-batch gradient descent we used. Some mini-batches have randomly unlucky data for optimization that causes exploding gradients that leads to spikes. It may also due to outliers in the data. Another abnormality is we have some value if accuracy jump to 0 because we have big goals that the model is not able to to accept.

5.2 GPT-NEO-Math

In the GPT-NEO-MATH, we ran into small problems initially. The size of the model was really large even for large RTX GPU with 48GB of RAM. We couldn't only try for the smallest GPT-NEO model. This means that we have left a lot of possible performance on the table. Although the Validations loss function described in Figure 44, has flattened out, it would be really interesting to look for greater number of epochs. The speed of training was very slow. We used Microsoft DeepSpeed to fasten our performance but we saw minimal improvements. We tried other accelerators including Huggingface accelerate but saw no speed improvements.

5.3 GPT-NEO-ReRanker

With Minimal training, we could see good results for ReRanking. Average number for finding the best tactic was within 6 tries. One possible area of improvement would be to train the GPT-NEO-ReRanker with more negative results to reduce the chance of possible false positives.

5.4 Difference between different Math Topics

One of the key takeaways was that the final score of different GPT models does not show a correlation with the dataset size.

References

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink training of BERT rerankers in multi-stage retrieval pipeline. *CoRR*, abs/2101.08751.

Dataset name	GPT-3 score	GPT-neo score	GPT combine score
Algebra	0.494	0.331	0.573
Algebraic Geometry	0.446	0.304	0.47
Algebraic Topology	0.479	0.362	0.545
Analysis	0.319	0.286	0.42
Category theory	0.5	0.252	0.549
Combinatorics	0.332	0.269	0.386
Computability	0.513	0.483	0.65
Control	0.62	0.331	0.707
Data	0.387	0.358	0.502
Deprecated	0.443	0.271	0.46
Dynamics	0.223	0.153	0.269
Field_theory	0.259	0.305	0.382
Geometry	0.316	0.202	0.391
Group_theory	0.334	0.324	0.431
Linear_algebra	0.303	0.254	0.407
Logic	0.452	0.291	0.527
Measure_theory	0.347	0.267	0.425
Number_theory	0.325	0.282	0.429
Order	0.336	0.303	0.435
Probability_theory	0.206	0.191	0.277
Ring_theory	0.292	0.288	0.397
Set_theory	0.335	0.331	0.486
System	0.259	0.314	0.336
Tactic	0.524	0.463	0.651
Testing	0.469	0.594	0.594
Topology	0.302	0.291	0.406

Table 1: Models accuracy percentage by topics for 3 different models

Ruiqi Guo, Quan Geng, David Simcha, Felix Chern, Sanjiv Kumar, and Xiang Wu. 2019. New loss functions for fast maximum inner product search. *CoRR*, abs/1908.10396.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *CoRR*, abs/2105.00377.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. 2022. Formal mathematics statement curriculum learning. *CoRR*, abs/2202.01344.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, and Susannah Young. 2021. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446.

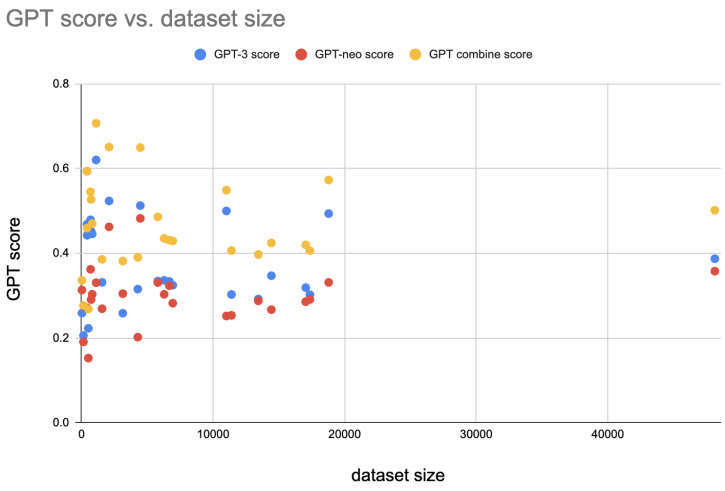

Figure 2: Scatter plot between model performance and dataset size

Figure 3: Train and validation loss curve for GPT-3-ADA

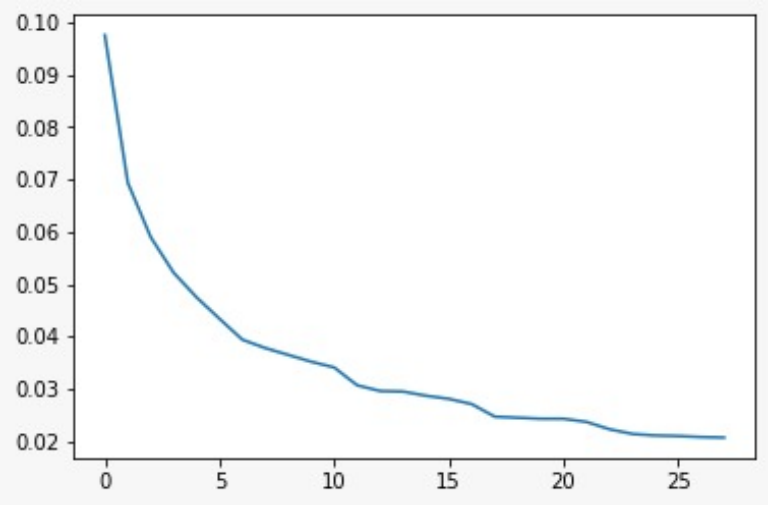

Figure 4: Validation loss curve for GPT-3-NEO