Project Proposal: Transformer-based Model for Mathematical Problem Solving

Chau Nguyen
chauminh.nguyen@mail.utoronto.ca

Gabriel You
gabriel.you@mail.utoronto.ca

Takia Talha
takia.talha@mail.utoronto.ca

Taha Siddiqi
taha.siddiqi@mail.utoronto.ca

Department of Mathematical & Computational Sciences
University of Toronto Mississauga

Abstract

Mathematical problem solving often requires both symbolic manipulation and natural language understanding. In this paper, we propose a hybrid, lightweight, and open-source model that combines symbolic mathematics with deep learning to solve mathematical problems expressed in natural language or LaTeX format. Our approach integrates several powerful components: `LaTeX2SymPy` for parsing mathematical expressions from LaTeX, `MathBERT` for tokenizing and encoding both natural language and symbolic math content, `SymPy` for symbolic computation and exact mathematical solving, and a Seq2Seq decoder for generating natural language output from symbolic results. This architecture allows the model to handle complex mathematical reasoning tasks, from simple algebraic equations to advanced calculus, while providing accurate and interpretable solutions. The model is efficient, scalable, and entirely open-source, enabling easy integration into educational tools, computational software, and research applications. We demonstrate that our hybrid model leverages the strengths of both symbolic and neural approaches to deliver a more robust solution for automated math problem-solving.

1 Introduction

A challenge often faced by students in the computational sciences is learning how to solve logically intensive math questions. Often times the transition from computation focused math to reasoning focused math presents a large learning curve due to the difficulty of developing mathematical intuition and a lack of rigorous step by step answers to compare with. We hope to develop an accessible model that can take in a latex text input of a math problem and output a descriptive and accurate latex text answer output to the problem. Developing models capable of solving this task is an infamously difficult problem **?** due to the requirement of consistency and mathematical riqour within the answers outputted by the models.

Currently the best models in this area use transformer LLM archetictures with large pretraining datasets. The downsides of these models is that the consistency may be poor due to multiple ways to present the same problem which are treated differently by the model. To solve for this we are incorporating a symbolic model into a transformer LLM architecture to hopefully increase accuracy within our math solving model. We believe this deep learning approach is reasonable as

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

understanding the language of math problems is something suited for transformer LLM models and creating consistency in mathematical reasoning is something that symbolic models are good at. By combining these 2 approaches we hope that it will combine the best of both architectures. With this work we hope to develop a more robust and helpful model that is able to answer with reason and provide thorough feedback on math questions potential users might have.

2 Background and Related Work

The application of large language models (LLMs) such as BERT **?** and GPT **?** has revolutionized the field of natural language processing (NLP). These transformer-based models excel at a wide range of tasks, including text classification, question answering, and translation. However, when it comes to mathematical reasoning and algebraic problem solving, LLMs often face challenges due to their primary design focusing on natural language rather than symbolic or mathematical computation.

To address this, there has been increasing interest in integrating symbolic reasoning capabilities into neural architectures. Symbolic models are traditionally used for tasks that require formal manipulation of symbols, such as equation solving, logical inference, and symbolic differentiation. Recent efforts have focused on combining the power of neural networks with symbolic reasoning systems to enhance the models' performance on tasks that involve mathematics or formal logic.

2.1 The Role of Symbolic Models in Mathematical Reasoning

Symbolic models have long been a central tool in areas such as algebra, calculus, and theorem proving. These models represent mathematical objects (e.g., variables, equations, operators) explicitly and can apply algebraic rules, differentiation, integration, and other symbolic operations directly to manipulate these objects. However, traditional symbolic approaches face scalability issues when dealing with large, unstructured datasets, such as those encountered in real-world mathematical problem solving.

On the other hand, neural networks, particularly deep learning models, excel at processing unstructured data like natural language and images. Yet, they often struggle with the step-by-step logical reasoning required for tasks like algebraic equation solving or symbolic manipulation due to their inability to explicitly model mathematical operations. This limitation has motivated research into hybrid approaches that combine the benefits of both symbolic models and neural architectures.

2.2 Integration of Symbolic Models into Transformer-based Architecture

Recent work has explored the integration of symbolic reasoning within Large Language Models, particularly in the context of mathematical problem solving. A key idea is to augment traditional neural models with external symbolic components that can handle explicit algebraic operations. For example, incorporating *symbolic equation solvers* into a Generative Pretrained Transformer (GPT) allows neural networks to handle more complex algebraic problems **?**. However, GPTs and other Large Language Models are large and not open-source, which makes it challenging for nonprofit educational organizations to conduct research and deploy it effectively due to limited access, high computational costs, and restrictions on customization.

On the other hand, Bidirectional Encoder Representations from Transformers (BERT), originally designed for NLP tasks, has been shown to benefit from fine-tuning on an extensive and diverse mathematical corpuses. Devlin et al. (2024) proposed the finetuning of BERT on mathematical datasets to better understand and solve algebraic problems **?**. Given that BERT and its variants, including mathBERT, are open source, lightweight and accessible to the general public, research and development in extending its current performance would be more easily motivated and conducted. However, to handle more complex mathematical reasoning tasks that involve symbolic manipulation, BERT-based models can be augmented with symbolic reasoning modules that perform operations like solving equations, simplifying expressions, and computing symbolic derivatives. This approach attempts to overcome the limitations of pure neural models by integrating symbolic computations into the reasoning process of the model.

Recent advances have focused on combining symbolic models with large language models to improve their ability to solve algebraic problems. In the context of BERT and similar models, symbolic models can be incorporated into the encoder-decoder structure, either by pre-processing mathematical inputs

with symbolic solvers or by augmenting the model's latent space with symbolic representations that guide its reasoning. For example, the work by Ünsal et al. **?** introduces a method where symbolic expressions produced by symbolic models are integrated directly into the attention mechanism of transformer-based models, allowing the network to learn from an accurate chain of subexpressions as part of the mathematical solving procedure as if they were a a paragraph composed of sentences.

2.3 Contributions of the Current Work

In this work, we propose a novel approach to algebraic problem solving by incorporating a symbolic model into the BERT encoder. We aim to enhance the BERT architecture with an external symbolic solver that can manipulate algebraic expressions, perform equation solving, and reason about mathematical objects symbolically. By integrating these symbolic capabilities directly into the BERT encoder, we hypothesize that the model will achieve improved performance on algebraic problem-solving tasks, enabling it to generate both correct and interpretable solutions to complex algebraic problems. Our work builds on recent advances in symbolic-augmented language models and aims to push the boundaries of what is achievable in algebraic reasoning within LLMs.

3 Data

3.1 Datasets

For this project, we decided to use the following datasets:

Math Dataset

The MATH dataset **?** consists of 12,500 (7,500 training and 5,000 test) problems from mathematics competitions including the AMC 10, AMC 12, AIME, and more. Many of these problems can be collected from AoPS Contests. **?**

NuminaMath-CoT

The Numina Math CoT dataset has approximately 860k math problems, where each solution is formatted in a Chain of Thought (CoT) manner. The sources of the dataset range from Chinese high school math exercises to US and international mathematics olympiad competition problems. **?**

3.2 Data Formatting

Problems and solutions are formatted using LATEX. The usage of LATEX ensures that the data is easily readable and is easy to parse and process for training the model.

The data for the MATH Dataset is formatted as a JSON file, with each problem containing the following fields:

- problem: The text of the problem
- level: The difficulty level of the problem (Level 1 up to Level 5)
- type: The type of math problem (e.g. algebra, geometry, etc.)
- solution: The solution to the problem

The data for the Numina Math CoT dataset are formatted as a JSON file, with each problem containing the following fields:

- An array of objects, where the first object contains
  - content: The text of the problem
  - role: the role assigned to the person who can access this data (user)
- The second object contains
  - content: The solution to the problem
  - role: the role assigned to the person who can access this data (assistant)

3.3 Data Preprocessing

We converted the data into the following format:

- problem: The text of the problem
- solution: The solution to the problem

In this way, the features and labels are clearly defined, making it easier to train the model.

Then, we filtered all foreign characters and symbols (i.e. Chinese characters) from the data to ensure that the model is trained on clean and consistent data. This preprocessing step helps to remove any noise or irrelevant information that could affect the model's performance.

Finally, we tokenized the text data using the MathBERT tokenizer, which is specifically designed for mathematical tasks. This tokenizer splits the input into subword tokens that represent both natural language components and mathematical symbols, enabling the model to process math-related queries effectively.

3.4 Data Splitting

We split the data into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune hyperparameters and prevent overfitting, and the test set is used to evaluate the model's performance on unseen data.

The split is as follows:

- Training set: 80%
- Validation set: 10%
- Test set: 10%

We distributed the Numina CoT dataset across the training, validation, and test sets. However, since the MATH dataset only had 12.5k problems, it was merged with the test set to evaluate the model's performance on a different set of problems.

4 Model Architecture

The proposed model aims to combine the strengths of symbolic math-solving with neural language understanding by utilizing a hybrid architecture that integrates `LaTeX2SymPy` for parsing mathematical expressions, the `mathBERT` tokenizer and encoder for understanding the natural language problem, `SymPy` for symbolic manipulation and computation, and a Seq2Seq decoder for generating textual output. The architecture is designed to be lightweight, efficient, and open-source, making it accessible for deployment in a wide range of applications.

4.1 Input Layer: LaTeX2SymPy Parsing

The input to the model is a natural language mathematical problem, often expressed in LaTeX format. To handle the structured mathematical expressions present in LaTeX format, we use `LaTeX2SymPy`. This component parses the LaTeX input into a symbolic expression that can be processed by the subsequent layers.

- **LaTeX Parsing:** Convert LaTeX-based input strings into tokenized symbolic expressions.
- **Symbolic Representation:** Map the parsed LaTeX expressions into `SymPy` objects, which represent mathematical terms, operators, and functions in a format suitable for algebraic manipulation.

This parsing step ensures that the mathematical problem is represented in a structured and mathematically accurate form, making it ready for further symbolic operations.

4.2 SymPy Solver

Once the problem is tokenized and embedded by the MathBERT encoder, it is passed to the `SymPy` Solver for symbolic computation. `SymPy` is a Python library dedicated to symbolic mathematics, and its role in this architecture is twofold:

- **Symbolic Manipulation:** `SymPy` is used to perform algebraic simplifications, differentiation, integration, solving equations, and other symbolic manipulations based on the parsed mathematical expression. This step leverages the full symbolic capabilities of `SymPy` to compute exact solutions when possible.
- **Mathematical Inference:** For problems requiring a numerical solution (such as solving equations), `SymPy` can use numerical solvers to provide an approximate answer. `SymPy` supports a wide range of operations, ensuring that the model can handle complex algebraic, calculus, and number-theoretic problems.
- **Generation of Subexpressions** The SymPy solver processes a given function one step at a time. To simulate the process of arriving at a mathematical solution, we feed the given expression into SymPy, then repeatedly feed the output expression into the SymPy until a solution is arrived. The generated subexpressions are sequentially concatenated in the order that they are produced, and the final string is passed into the tokenizer. The goal is for the model to not only learn the positional embeddings and the underlying relations behind the word and mathematical expression parts of a given problem, but also to learn the explicit steps leading up to the solution.

This integration ensures that the model not only understands the mathematical structure of the problem but also can derive precise and reliable solutions when applicable.

4.3 MathBERT Tokenizer and Encoder

The parsed mathematical expression is fed into a `MathBERT` tokenizer and encoder, which is a specialized version of BERT fine-tuned for mathematical tasks **?**. MathBERT has been trained to handle both natural language and symbolic mathematical notation, which allows it to effectively process math-related queries and understand the underlying structure of the problem. The steps involved are as follows:

- **Tokenization:** The `MathBERT` tokenizer splits the input into subword tokens that represent both natural language components and mathematical symbols. This step is crucial for handling mixed inputs such as text-based questions and mathematical formulas.
- **Contextual Embedding:** The tokenized input is passed through the `MathBERT` encoder, which generates contextual embeddings for each token. These embeddings capture semantic relationships between mathematical operations, variables, and their corresponding natural language descriptions. The encoder's attention mechanism enables the model to focus on relevant parts of the input when generating solutions.

4.4 Seq2Seq Decoder

The final step of the architecture involves the use of a Seq2Seq decoder, which generates the natural language output from the symbolic solution obtained from `SymPy`. The decoder is designed to convert symbolic results back into human-readable text, making the solution interpretable and usable for end-users.

- **Input:** The output from `SymPy` (whether symbolic or numeric) is used as input to the decoder.
- **Generation:** The Seq2Seq model is trained to map the symbolic solution into a sequence of natural language words or sentences. This component ensures that the model can output the solution in an understandable and grammatically correct format.

This step integrates both symbolic manipulation and natural language generation, ensuring that the solution is not only mathematically accurate but also linguistically appropriate.

4.5 Advantages of the Proposed Architecture

The hybrid architecture offers several advantages:

- **Hybrid Approach:** The combination of symbolic computation (via `SymPy`) and deep learning (via `MathBERT` and Seq2Seq) leverages the strengths of both methods, enabling the model to handle complex mathematical reasoning and natural language understanding simultaneously.
- **Lightweight and Efficient:** The model is designed to be efficient by using pre-trained BERT embeddings and leveraging `SymPy` for exact symbolic computation, reducing the need for large-scale numerical solvers or extensive training data.
- **Open Source:** All components of the model (`SymPy`, `MathBERT`, Seq2Seq, and `LaTeX2SymPy`) are open-source, ensuring transparency, reproducibility, and accessibility.
- **Scalability:** The model can handle a wide variety of mathematical problems, from basic algebra to more advanced calculus and number theory, making it suitable for a range of educational and professional applications.

5 Model architecture figure

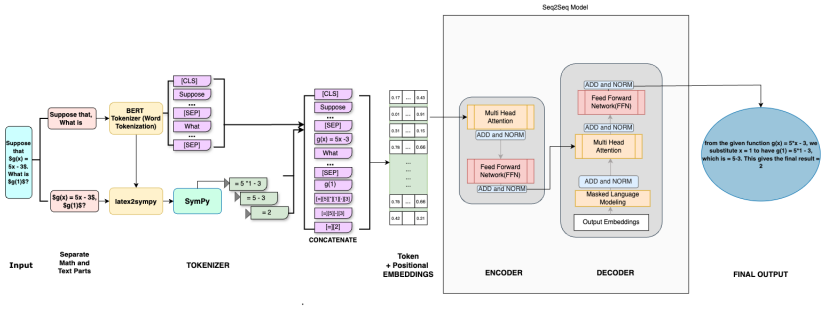

Figure 1: The Transformer - model architecture.

6 Results

7 Discussion

8 Limitations

9 Ethical considerations

9.1 Ethical Issues

One ethical consideration is the potential for the model to be used to cheat on math assignments/homework. This could encourage students to use the model as a shortcut rather than engaging with the material themselves, which would negatively impact a students capacity for critical thinking and problem-solving.

Another ethical issue is intellectual property rights, as the model is trained on copyrighted data. The model could inadvertently promote plagiarism if it directly provides solutions that students submit as

their own, without understanding the learning process. This could lead to academic dishonesty and undermine the integrity of the educational system.

9.2 Mitigation Strategies

To mitigate the first risk, the model should be used as a learning tool rather than a tool for cheating. For example, the model could be used to generate practice problems for students to solve, or to provide explanations for the solutions to problems. This would help students learn and improve their math skills, rather than using the model to cheat. Therefore, while automation can support learning, it is essential that it complements, rather than replaces, active engagement with the educational process.

To mitigate the second risk, the model should be used in a controlled environment where students are guided on how to use the model appropriately. For example, teachers could provide guidelines on how to use the model to check answers or generate practice problems, rather than using it to directly provide solutions. Furthermore, it is crucial to emphasize the value of understanding the material and using tools as aids rather than shortcuts. This would prevent plagiarism and encourage students to engage with the material and develop their problem-solving skills.

10 Conclusion