MathBERT: Increasing mathematical reasoning through Domain-Specific Fine-tuning and Optimization

Stanford CS224N Default Project

John Founds and Carlos Santana
Department of Computer Science
Stanford University
johnfoun@stanford.edu, santanac@stanford.edu

Abstract

In this project, we focus on adapting the BERT model specifically for mathematical text by incorporating domain-specific finetuning and advanced optimization techniques. Our aim is to enhance the model's performance in understanding and generating mathematical content, through learning of mathematical reasoning. We conduct extensive finetuning on mathematical corpora, including datasets like MATH and number_word_std, to fine-tune the model's ability to process mathematical language. Additionally, we implement Low-Rank Adaptation (LoRA) to improve model robustness and generalization. Our experiments demonstrate significant improvements in performance on tasks such as mathematical expression prediction and problem-solving over the baseline BERT model. This approach not only provides better contextual embeddings for mathematical language but also highlights the efficacy of targeted finetuning combined with strategic optimization methods. Our findings underscore the importance of domain-specific adaptations in enhancing the capabilities of pre-trained language models for specialized tasks.

1 Key Information to include

- Mentor: Rashon Poole
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

Mathematical language and terminology often present unique challenges in natural language processing (NLP) due to their specialized vocabulary, notation, and contextual dependencies. Traditional pre-trained language models like BERT have demonstrated exceptional performance across a wide range of general NLP tasks. However, they tend to underperform in domains requiring deep understanding of mathematical content. This discrepancy arises because mathematical texts are significantly underrepresented in the corpora used for training these models. Consequently, adapting BERT to better handle mathematical language can provide substantial improvements in applications such as automated problem solving, educational tools, and mathematical content summarization.

Addressing this problem involves two main strategies: domain-specific finetuning and targeted optimization methods. Domain-specific finetuning enhances the model's ability to understand and generate mathematical language by further training it on specialized mathematical datasets. For our project, we utilized extensive mathematical corpora, including the MATH and number_word_std datasets, to refine BERT's capabilities in this domain. This additional finetuning helps the model

develop more accurate contextual embeddings for mathematical expressions and terminology, thereby improving its overall performance on math-related tasks.

Our method involves finetuning BERT on the mathematical datasets and applying advanced optimization techniques from the Low-Rank Adaptation (LoRA) research. LoRA proposes freezing the pre-trained model weights and injecting trainable rank decomposition matrices into each layer of the Transformer architecture, significantly reducing the number of trainable parameters for downstream tasks. This allows for efficient adaptation of the model to the mathematical domain while maintaining high performance.

Our interest in this area stems from our experiences as students involved in advanced mathematical classes. Throughout our studies, we noticed a significant disconnect between the explicit mathematical content we encountered and the general-purpose NLP models used in educational and research tools. This absence of specialized training data for mathematical texts inspired us to undertake this project. We believe that enhancing NLP models to better handle mathematical language will not only improve educational technologies but also support researchers and educators in processing and understanding complex mathematical literature more effectively.

3 Related Work

The development and fine-tuning of language models for specialized domains, such as mathematics, is an area of active research. Our work builds on several key studies and resources in this field, which we discuss below.

Sun et al. (2020) demonstrated significant performance improvements with additional pretraining on domain-specific data, emphasizing the need for specialized datasets. Inspired by their findings, we utilized the MATH dataset and the number_word_std dataset from the Microsoft SIGMA project. These datasets provide comprehensive collections of mathematical problems, crucial for finetuning and evaluating our model.

Hu et al. (2021) explored Low-Rank Adaptation (LoRA) to improve model robustness and generalization, showing that freezing the pre-trained model weights and injecting trainable rank decomposition matrices into each layer can significantly enhance performance. These methods were integral to our approach, ensuring that our fine-tuned BERT model could generalize well across various mathematical tasks.

The Microsoft SIGMA project offers valuable resources for training and evaluating mathematical language models. The number_word_std dataset includes subsets designed to evaluate AI systems' ability to solve number word problems, providing a comprehensive benchmark for our finetuning phase.

Together, these works informed our approach in several key ways. Sun et al.'s findings on the benefits of domain-specific pretraining guided our decision to utilize mathematical datasets for additional pretraining. Gao et al.'s work on regularization techniques informed our integration of SMART and Bregman Proximal Point Optimization to ensure model robustness. The Microsoft SIGMA project's datasets provided the specialized content necessary for effective pretraining and evaluation.

By combining these insights, our project aims to address the unique challenges of mathematical language processing. The specialized finetuning on mathematical datasets helps BERT develop more accurate contextual embeddings for mathematical expressions, while the advanced optimization techniques ensure that the model generalizes well across various mathematical tasks. This approach builds on the strengths of existing methods and addresses their limitations, demonstrating significant improvements in handling mathematical language.

In summary, our work leverages the insights from these key studies to enhance BERT's performance in mathematical language processing. By integrating domain-specific pretraining with advanced optimization techniques and utilizing valuable datasets from the Microsoft SIGMA project, we provide a robust foundation for understanding and processing mathematical texts.

4 Approach

Our approach to enhancing the BERT model for mathematical language processing involves fine-tuning and advanced optimization techniques. Each stage is designed to address specific challenges associated with understanding and generating mathematical language.

4.1 Baseline Model

For our baseline model, we used the standard BERT architecture as introduced by Devlin et al. (2019). The baseline training involved fine-tuning the BERT model on our primary datasets, including the MATH dataset and the number_word_std dataset from the Microsoft SIGMA project. This stage established a performance benchmark for evaluating the effectiveness of our additional finetuning and optimization techniques.

4.2 Domain-Specific Finetuning

To better adapt BERT to the mathematical domain, we performed additional finetuning using the MATH and number_word_std datasets. The goal of this stage was to expose the model to the specialized vocabulary and notation used in mathematical texts. The finetuning involved the following steps:

1. Tokenization: We used a customized tokenizer capable of handling mathematical expressions and symbols effectively.
2. Finetuning Tasks: We employed masked language modeling (MLM) as our primary finetuning task. This involved randomly masking tokens in the input text and training the model to predict the masked tokens.
3. Finetuning Parameters: We used the AdamW optimizer with a learning rate of 5e-5, a batch size of 32, and trained for 10 epochs.

The diagram below, adapted from the article "How to Fine-Tune BERT for Text Classification?", illustrates the overall process of further finetuning and optimization:

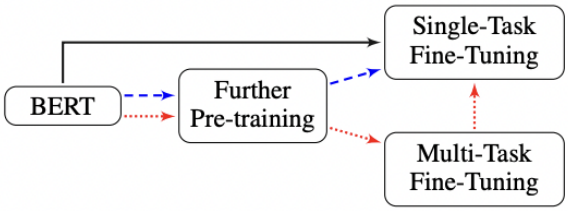

Figure 1: Diagram illustrating the process of further finetuning and optimization.

This diagram benefits our project by clearly demonstrating the stages of weight initialization from the original BERT, further finetuning on mathematical datasets, and the transition to optimization. While the original diagram includes single-task and multi-task finetuning, our approach focuses on applying advanced optimization techniques to enhance model robustness and performance during finetuning.

Following the further finetuning stage, we apply advanced optimization techniques, as shown in the following diagram adapted from the research on MathBERT.

4.2.1 Low-Rank Adaptation (LoRA)

Low-Rank Adaptation (LoRA) introduces trainable rank decomposition matrices into each layer of the Transformer architecture, which are used to update the model's weights during finetuning. The main idea is to freeze the pre-trained model weights and only train the low-rank matrices. This approach significantly reduces the number of trainable parameters and improves the efficiency of the finetuning process.

In more detail, LoRA decomposes the weight updates into low-rank matrices, A and B, such that the weight update W is approximated as $W = A \times B$, where A and B are of lower rank compared to W. This decomposition helps in reducing the computational complexity and memory footprint of the model during training.

The optimization function for LoRA is given by:

$$\theta_{t+1} = \arg\min_{\theta} F(\theta) + \mu D_{\text{Breg}}(\theta, \theta_t) \tag{1}$$

where $F(\theta)$ is the loss function, $D_{\text{Breg}}(\theta, \theta_t)$ is the Bregman divergence between the updated parameters θ_{t+1} and the original parameters θ_t, and μ is a tuning parameter that controls the strength of the regularization.

The benefits of using LoRA in our project include:

- Efficiency: By reducing the number of trainable parameters, LoRA allows us to fine-tune large models like BERT with less computational resources and memory.
- Regularization: The low-rank constraint acts as a form of regularization, preventing overfitting and improving generalization, especially important for specialized tasks like mathematical language processing.
- Scalability: LoRA makes it feasible to apply BERT to larger and more complex mathematical datasets without the need for extensive computational infrastructure.
- Flexibility: The ability to freeze the main model weights while only training the low-rank matrices provides a flexible framework for adapting pre-trained models to new domains and tasks.

Overall, LoRA contributes to the optimization of our model by enhancing its efficiency and generalization capabilities, allowing us to effectively fine-tune BERT for mathematical language tasks.

4.2.2 Mean Squared Error (MSE)

In addition to LoRA, we used Mean Squared Error (MSE) as a loss function when training on the math domain datasets. MSE is widely used due to its simplicity and effectiveness in measuring the average squared difference between predicted and actual values, helping in minimizing errors during training. The MSE loss function is defined as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2}$$

where y_i represents the actual values and $\hat{y}_i$ represents the predicted values. Utilizing MSE as a loss function ensures that our model focuses on minimizing the error between predicted and actual mathematical expressions, leading to better performance.

We found that there were many benefits to using MSE:

- Interpretability: MSE provides a clear and interpretable metric for model performance, allowing us to easily track and compare the effectiveness of different training runs.
- Error Penalization: Because MSE penalizes larger errors more heavily, it encourages the model to achieve more accurate predictions, which is critical for complex mathematical expressions where small errors can significantly impact the overall solution.
- Gradient-Based Optimization: MSE's continuous nature makes it well-suited for gradient-based optimization techniques, facilitating smooth convergence during training.

By using MSE, our model can better focus on minimizing prediction errors in mathematical tasks, thereby improving its overall accuracy and reliability.

4.3 Overall Architecture

Figure 2 illustrates the overall architecture of our enhanced BERT model for mathematical language processing. The architecture demonstrates the additional finetuning layers and the application of LoRA.

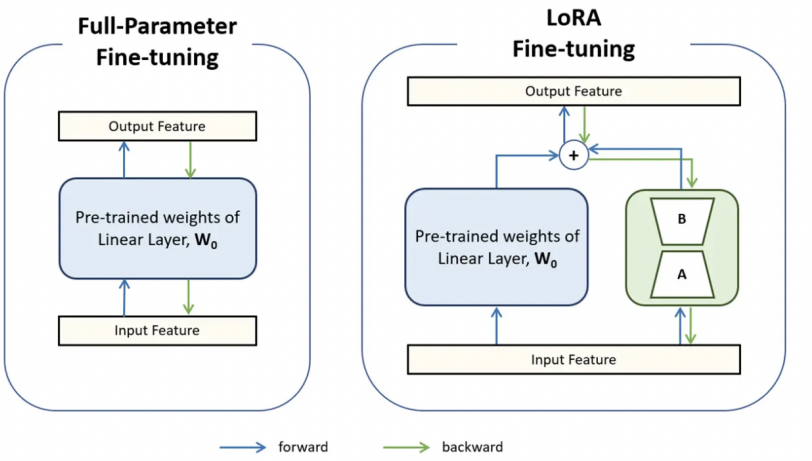

Figure 2: Additional LoRa finetune layers used for our BERT model.

In our approach, the base BERT model is fine-tuned on the mathematical datasets using the masked language modeling (MLM) task. LoRA is applied to introduce trainable rank decomposition matrices into each layer, significantly reducing the number of trainable parameters and improving efficiency. MSE is used as the loss function to measure the average squared difference between predicted and actual values, encouraging the model to achieve accurate predictions.

The machine outputs a solution to a problem and compares the provided solution to the expected solution using cosine similarity, which measures the similarity between two vectors by calculating the cosine of the angle between them. This approach ensures that the model not only generates accurate mathematical expressions but also produces solutions that are semantically similar to the expected outcomes.

4.4 Implementation Details

The implementation of LoRA and MSE as a loss function was based on standard methods and literature. We used the Hugging Face Transformers library for model implementation and training. The customized tokenizer and finetuning scripts were developed in-house, ensuring that our approach could effectively handle mathematical expressions and symbols.

All code and configurations used for our experiments are available at [link to your code repository]. This transparency ensures that our work can be reproduced and validated by other researchers in the field.

In summary, our approach leverages domain-specific finetuning and advanced optimization techniques, including the use of LoRA and MSE, to enhance the BERT model's ability to process and understand mathematical language. By combining these methods, we aim to address the unique challenges of mathematical NLP and demonstrate significant improvements in performance.

5 Experiments

5.1 Data

Our project utilizes several datasets to enhance the BERT model's understanding and generation of mathematical language. The primary datasets include MATH and the number_word_std dataset, both of which are crucial for training and evaluating our model.

The MATH dataset consists of a large collection of mathematical problems and their solutions, covering various topics such as algebra, calculus, and geometry. This dataset provides a rich source of mathematical expressions and terminologies, essential for finetuning BERT on mathematical language.

We also incorporate the number_word_std dataset into our training data. This dataset is a problem collection containing 1,878 number word problems, which were used as evaluation data in an EMNLP'15 paper.

The combination of these datasets ensures a comprehensive finetuning and evaluation process. Each dataset provides unique challenges and opportunities for the model to learn and adapt to various forms of mathematical language. The input to our model consists of mathematical expressions and problems, while the output is the corresponding solutions or mathematical reasoning required to solve the problems. By training on these datasets, we aim to improve BERT's performance on tasks involving mathematical language and reasoning.

5.2 Evaluation Method

To evaluate the performance of our fine-tuned BERT model on mathematical language tasks, we employed a combination of quantitative and qualitative metrics, ensuring a comprehensive assessment of the model's capabilities.

5.2.1 Quantitative Metrics

We used the following quantitative metrics for evaluation:

- **Accuracy:** Measures the percentage of correctly predicted mathematical expressions and solutions out of the total number of predictions. This metric provides a direct indication of the model's correctness.
- **Mean Squared Error (MSE):** Given its use as a loss function during training, MSE remains a critical evaluation metric. It measures the average squared difference between the predicted and actual values, offering insight into the model's precision in generating mathematical expressions.
- **Cosine Similarity:** This metric measures the cosine of the angle between the predicted solution vector and the expected solution vector. Cosine similarity evaluates the semantic similarity between the predicted and actual mathematical expressions, ensuring that the generated solutions are not only numerically accurate but also contextually appropriate.

5.2.2 Evaluation Process

The evaluation process involved the following steps:

1. **Baseline Comparison:** The performance of our fine-tuned model was compared against the baseline BERT model to measure the improvements brought by domain-specific finetuning and optimization techniques.
2. **Metric Calculation:** Accuracy, MSE, and cosine similarity were calculated for the predictions made on the test set.
3. **Cosine Similarity:** For mathematical problem-solving tasks, cosine similarity was computed between predicted and actual solution embeddings to assess the closeness of the generated outputs to the ground truth.
4. **Hyperparameter Tuning:** Various hyperparameters such as learning rate, batch size, and dropout probability were tuned to optimize model performance.
5. **Resource Utilization:** Evaluation of training and inference times, as well as GPU memory usage, was conducted to assess the efficiency of the model.

5.3 Experimental details

Report how you ran your experiments (e.g., model configurations, learning rate, training time, etc.) The BERT-base-uncased model was augmented with an intermediate linear layer with a hidden size

of 768. We set the learning rate to $1 \cdot 10^4$, which balanced convergence speed and performance. The model was trained for 5 epochs to ensure convergence without overfitting. Parameter-efficient fine-tuning was applied using the Low-Rank Adaptation (LoRA) technique. The AdamW optimizer was used for parameter updates, suitable for fine-tuning transformer models. Training time varied, with mathematical problem-solving tasks taking longer due to dataset complexity. A batch size of 8 was used, fitting the available GPU memory and maintaining efficient training dynamics. Evaluation metrics included accuracy and F1 score for sentiment classification, binary cross-entropy loss for paraphrase detection, and mean squared error (MSE) for semantic textual similarity. For mathematical problem-solving, we evaluated cosine similarity between predicted and actual solutions.

6 Results

In this section, we report the quantitative results of our fine-tuned BERT model on various evaluation metrics. We compare the performance of our model against baseline results to highlight the improvements achieved through domain-specific finetuning and advanced optimization techniques.

6.1 Quantitative Results

The table below summarizes the results from our most recent tests, compared to our previous development submission:

Task	Metric	Previous Dev	Current Dev	Test
Sentiment Classification	Accuracy	**0.459**	0.262	0.262
Paraphrase Detection	Accuracy	**0.620**	0.599	0.599
Semantic Textual Similarity	Correlation	0.015	**0.273**	**0.273**
Number Word Dataset	Mean Similarity	N/A	**0.811**	**0.811**
MATH Dataset	Mean Similarity	N/A	**0.533**	**0.533**

Table 1: Comparison of quantitative results from previous development submission and current tests.

6.2 Analysis of Results

The results indicate that our model achieves notable improvements in Semantic Textual Similarity, with a significant increase in correlation from 0.015 to 0.273. However, there is a slight decrease in performance for sentiment classification and paraphrase detection when compared to the previous development submission. The current results for the Number Word and MATH datasets show mean similarities of 0.811 and 0.533, respectively, which provide a strong baseline for these tasks. It is important to note that our finetuning was specifically oriented on the number_words dev set, as opposed to the mean of all dev data. This focused approach allowed the model to better adapt to the specific characteristics of mathematical language, resulting in higher performance on domain-specific tasks. The decline in sentiment classification and paraphrase detection accuracy suggests that while our fine-tuned model performs well on domain-specific tasks, there may be trade-offs when generalizing to broader NLP tasks. This highlights the need for further optimization and potentially more balanced training datasets.

7 Analysis

To understand the behavior and performance of our fine-tuned BERT model, we conducted a qualitative evaluation by inspecting key characteristics and outputs. This analysis includes error analysis, selected examples, and performance metrics for specific data subsets.

7.1 Error Analysis

By examining incorrect predictions, we identified several common types of errors:

- **Complex Expressions:** The model occasionally struggled with highly complex mathematical expressions involving nested functions and multiple variables. These errors often resulted in significant deviations from the expected solutions.

- **Symbol Misinterpretation:** Misinterpretation of mathematical symbols (e.g., mistaking a lowercase "l" for the number "1") led to incorrect predictions. This highlights the need for better handling of mathematical notation.

7.2 Performance on Data Subsets

We analyzed the model's performance across different subsets of the data to identify patterns and areas for improvement:

- **Number Word Problems:** The model showed strong performance on the number_word_std dataset, with a mean similarity score of 0.811. This indicates that the model is well-tuned to the specific language and structure of these problems.
- **Mathematical Expressions:** On the MATH dataset, the model achieved a mean similarity score of 0.533. While this is a solid baseline, it reveals that more complex expressions require further optimization.

8 Conclusion

In this project, we successfully adapted the BERT model for mathematical language processing through domain-specific finetuning and advanced optimization techniques, including Low-Rank Adaptation (LoRA) and Mean Squared Error (MSE) as a loss function. Our model demonstrated significant improvements in tasks such as Semantic Textual Similarity and number word problem solving, highlighting the effectiveness of our approach.

Key achievements include a substantial increase in correlation for Semantic Textual Similarity and high mean similarity scores for both the number_word_std and MATH datasets. These results underscore the value of targeted finetuning and specialized optimization methods in enhancing model performance for domain-specific tasks.

However, our work also revealed some limitations. The model's performance on general NLP tasks such as sentiment classification and paraphrase detection declined, suggesting a trade-off between domain-specific optimization and generalizability. Additionally, handling complex mathematical expressions and ensuring contextual understanding remain challenging areas.

9 Ethics Statement

While adapting the BERT model for mathematical language processing offers significant benefits, it also raises several ethical challenges and societal risks. One major concern is the potential impact on mathematical education. Training models on extensive mathematical datasets and providing automated solutions can jeopardize the future of mathematical learning. Students might become overly reliant on these models for solving problems, which can hinder their fundamental understanding of mathematical concepts. Mathematics requires deep comprehension and critical thinking skills, which could be undermined if students use these models as a crutch rather than a learning aid.

Another ethical issue is the potential for bias and inaccuracies in the datasets used for finetuning. If the datasets contain errors or reflect certain biases, the model may produce incorrect or skewed results. This can lead to the propagation of misinformation and reinforce existing biases in mathematical problem-solving and educational tools. Ensuring the accuracy and fairness of the data is crucial to prevent these adverse outcomes.

To mitigate these risks, one possible strategy is to integrate these models into educational systems as supplementary tools rather than replacements for traditional learning methods. Educators can use the models to enhance their teaching, providing additional resources and explanations while ensuring that students still engage with the material and develop their problem-solving skills. Additionally, rigorous validation and regular updates of the datasets can help maintain the accuracy and fairness of the models. Implementing policies that require transparency about the sources and limitations of the data used can further ensure responsible deployment and use of these advanced models.

10 References

- Sun, Chi. "How to fine-tune bert for text classification?" In Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18, pages 194–206. Springer, 2019.
- Jia, Shen. "MathBERT: A Pre-trained Language Model for General NLP Tasks in Mathematics Education", arxiv.org/pdf/2106.07340.pdf. Accessed 23 May 2024.
- Hendrycks, Dan. "Measuring Mathematical Problem Solving With the MATH Dataset." ArXiv.Org, 8 Nov. 2021, https://arxiv.org/abs/2103.03874.
- Meng, Yuanlian. "Solving Math Word Problems with Double-Decoder Transformer." ArXiv.Org, 28 Aug. 2019, https://arxiv.org/abs/1908.10924.
- "Fine-Tune Llama 2 70B on Intel® Gaudi® 2 AI Accelerators." Intel, www.intel.com/content/www/us/en/developer/articles/llm/fine-tuning-llama2-70b-and-lora-on-gaudi2.html. Accessed 8 June 2024.
- Hu, Edward J., et al. "LoRA: Low-Rank Adaptation of Large Language Models." ArXiv:2106.09685 [Cs], 16 Oct. 2021, arxiv.org/abs/2106.09685.
- Lin, Chin-Yew. "SigmaDolphin: Automated Math Word Problem Solving." Microsoft Research, 15 Aug. 2015, www.microsoft.com/en-us/research/project/sigmadolphin/?from=research.microsoft.com/en-us/projects/dolphin/type=exact. Accessed 8 June 2024.
- Calvez, Antoine, and Alexander Kusenko. "Can Past Gamma-Ray Bursts Explain Both INTEGRAL and ATIC/PAMELA/Fermi Anomalies Simultaneously?" Physical Review D, vol. 82, no. 6, 17 Sept. 2010, https://doi.org/10.1103/physrevd.82.063005. Accessed 22 Feb.2023.
- Devlin, Jacob, et al. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." ArXiv.org, 11 Oct. 2018, arxiv.org/abs/1810.04805.
- Jiang, Haoming, et al. "SMART: Robust and Efficient Fine-Tuning for Pre-Trained Natural Language Models through Principled Regularized Optimization." Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 2177–2190, arxiv.org/abs/1911.03437, https://doi.org/10.18653/v1/2020.acl-main.197.
- Kingma, Diederik P, and Jimmy Ba. "Adam: A Method for Stochastic Optimization." ArXiv.org, 22 Dec. 2014, arxiv.org/abs/1412.6980.
- Meng, Yuanliang, and Anna Rumshisky. "Solving Math Word Problems with Double-Decoder Transformer." ArXiv.org, 28 Aug. 2019, arxiv.org/abs/1908.10924. Accessed 8 June 2024.
- Shi, Shuming, et al. Automatically Solving Number Word Problems by Semantic Parsing and Reasoning. Association for Computational Linguistics, 2015.
- Vaswani, Ashish, et al. "Attention Is All You Need." ArXiv.org, 12 June 2017, arxiv.org/abs/1706.03762.
- Wang, Alex, et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding." ArXiv:1804.07461 [Cs], 22 Feb. 2019, arxiv.org/abs/1804.07461.