Creating a tool to solve math word problems: Math Solver

Abhinav Gupta[†] **Sutej Singh**[†]

University of Southern California, Los Angeles, CA, USA
[*]{abhinavg, sutejsin}@usc.edu

Abstract

This paper presents "Math Solver", a tool capable of solving intricate mathematical problems using advanced language models. Existing models, including GPT-4, show limitations in handling complex math word problems. Previous approaches, such as seq2seq models, reinforcement learning, and specialized architectures, have made progress but still fall short in accuracy.

The proposed project hypothesizes that employing newer and different language models, such as the transformer architecture, and fine-tuning models such as T-5 and GPT-2 can surpass this accuracy. The approach involves using corpora of math word problems, the GSM8K dataset from OpenAI. The model will generate not only the final numerical answer but also the logical steps leading to the solution. An evaluation tool will compare the similarity between these generated steps and those in the test set. The project's success will be measured by the accuracy of the final answer and the relevance of the procedural steps, aiming to significantly enhance the capabilities of language models in solving math word problems.

Introduction

Advanced language models like GPT-4 have revolutionized how we approach a broad spectrum of queries, offering insightful and contextually relevant responses across various domains. However, the realm of intricate mathematical problem-solving poses unique challenges. These challenges stem from the necessity to understand and manipulate numerical and symbolic information in a logically coherent manner, which is fundamentally different from general language processing.

Recognizing this gap, our project aims to develop a specialized tool that bridges the capabilities of advanced language models with the specific demands of mathematical reasoning. By incorporating elements of transformer architectures, known for their self-attention mechanism, this tool is designed to accurately weigh the relevance of different parts of a math problem. This is crucial for understanding the sequential and hierarchical nature inherent in mathematical logic.

Additionally, the use of transfer learning, where a model is pre-trained on a vast dataset of general text and then fine-tuned with mathematical language and problems, is a cornerstone of our approach. This methodology is expected to provide the model with a deep understanding of both natural and mathematical language, enabling it to parse, interpret, and solve math problems with an unprecedented level of accuracy.

The potential applications of such a tool are vast and varied:

1. Educational Aid for Students: It could serve as a revolutionary educational aid, helping students to understand and solve complex math problems. By providing not only solutions but also the logical steps and reasoning behind them, it could significantly enhance learning and comprehension.

2. Resource for Researchers: Researchers in fields requiring complex mathematical calculations could use this tool to overcome hurdles in their work, enabling more efficient and accurate problem-solving.

3. Tool for Educators: Educators could utilize this tool to develop solution keys and enrich problem sets, both existing and new. It could aid in curriculum development, allowing for the creation of more challenging and diverse problem sets that are accompanied by detailed solutions and explanations.

4. Assistance in Professional Fields: Professionals in fields like engineering, finance, and data analysis, where complex mathematical problem-solving is routine, could leverage this tool to streamline their workflows and enhance accuracy.

[‡]Equal contribution

Related Work

In the past, techniques like recurrent seq2seq models (Sutskever et al., 2014), reinforcement learning (Huang et al., 2018), specialized encoder-decoder architectures (Chen et al., 2020) have been used to approach this problem. OpenAI created a system based on GPT-3 (Cobbe et al., 2021) that was fine tuned for generative and verification tasks to solve grade level mathematical problems. The paper also introduced the GSM8K dataset, which is an improvement on other publicly available datasets in this domain since it contains detailed step-by-step solutions for over 8K math word problems. While these models are promising, even the highest performing implementation by OpenAI produces an accuracy of just 55%.

Hypothesis

Our hypothesis suggests that by using advanced Natural Language Processing (NLP) frameworks, especially those using transformer architectures, we can improve the way solutions to grade-level math problems are generated. Transformers are notable for their self-attention mechanism, which is key in processing the sequential and hierarchical aspects of math reasoning.

We also utilize transfer learning, where a model initially trained on a broad text corpus is then fine-tuned with specialized math language data. This approach is expected to not only enhance the accuracy of answers but also enable the model to explain the steps and logic involved in solving complex math problems.

Methodology

In our investigation into the automatic generation of solutions for math word problems, we leveraged OpenAI's GSM8K dataset along with three different models: 5-Gram, T5, and GPT-2—each chosen for their unique strengths and appropriateness for the task at hand. Below, we delve into the specifics of our methodology.

Dataset

The GSM8K dataset, sourced from OpenAI's open-source repository, is a comprehensive collection of 8,500 high-quality word problems, meticulously curated to represent a wide spectrum of mathematical concepts. This dataset is specifically designed to challenge and evaluate the capabilities of AI in understanding and solving complex mathematical problems typically encountered at a grade-school level.

Question	Answer
Emily makes $1,000,000 per year. If she has 10 employees who make $20,000 per year, how much would her salary be if she took part of her salary to make sure all of her employees make $35,000 per year?	Each employee needs $35,000-$20,000=$ «35000-20000=15000» 15,000 more per year. In total, Emily needs to pay $15,000*10= $«15000*10=150000» 150,000 extra. Emily will need to have a new salary of $1,000,000-$150,000= «1000000-150000=850000» 850,000 per year. #### 850,000

Table 1: Samples from the dataset

Each entry in the dataset is structured into two key components: a question and its corresponding answer. The questions are formulated to mimic real-world mathematical scenarios that students might encounter in an educational setting. They cover a diverse range of topics, from basic arithmetic to more advanced concepts, ensuring a comprehensive assessment of the AI's mathematical reasoning abilities.

The answer section of each entry is particularly noteworthy. It goes beyond providing a mere numeric answer; instead, it includes a detailed explanation of the logical reasoning required to solve the problem. This is crucial for our project, as it allows the AI not just to find the right answer, but to understand and replicate the thought process behind it. Mathematical steps involved in the solution are meticulously enclosed within angular brackets, making it easier for the model to distinguish between the narrative explanation and the actual mathematical calculations.

At the end of each answer, the numeric solution to the problem is distinctly marked, preceded by four pound signs (####). This unique formatting allows for easy identification of the final answer, facilitating the evaluation of the model's performance in terms of accuracy.

By splitting this dataset into training and test data, we have created an environment where our model can learn from a diverse array of problem types and complexities. The training data enables the model to understand and internalize the patterns, techniques, and logic involved in mathematical problem-solving. Meanwhile, the test data serves as a benchmark to evaluate the model's proficiency and its ability to generalize its learning to solve new, unseen problems.

5-Gram Model

The n-Gram model, a statistical language model, was employed as a baseline for our experiments. It operates on the principle of n-grams, where the probability of generating the next word depends only on the previous n words. We opted for a straightforward 5-gram model, utilizing OpenAI's dataset in JSON format. This simplicity makes the 5-Gram model computationally efficient and quick to train, which is why it served as an initial reference point. Although it lacks the sophistication of deep learning models, its usefulness lies in its speed and the straightforward nature of its predictions, providing a comparative framework against which the more complex models' improvements can be measured.

After determining the average answer length to be 48.10, we used this to set our word generation count. The training set's questions and answers were merged into a single string, upon which we trained the 5-gram model. We incorporated interpolation smoothing and normalized the resulting probabilities. As we generated each word, we updated the context to encompass the latest prediction, re-calibrating and normalizing the probability array for subsequent word generation. Using this setup, the 5-Gram model was trained rapidly, and it helped establish a foundational understanding of the problem's statistical structure.

T-5 Model (Text-to-Text Transfer Transformer)

T5 is a highly versatile and powerful model that redefines all NLP tasks into a unified text-to-text framework. It is pre-trained on a large corpus and can be fine-tuned for various tasks, including translation, summarization, and question answering. For generating math word problem solutions, T5's ability to understand context and generate coherent, extended text makes it particularly suitable. Its encoder-decoder architecture can handle the complex relationships and structures within math problems.

Question	Answer	Gen Answer
Leo and Agrey went fishing. Agrey caught 20 more fish than Leo. If Leo caught 40 fish, what's the total number of fish they caught together.	Agrey caught 40+20 = «40+20=60» 60 fish. The total number of fish they caught is 60+40 = «60+40=100» 100 fish. #### 100	together 20% to have 9 to each are and and to is a to was If is of are raise they school have are If much

Table 2: Sample output generated by 5-gram model

Training the T5 model requires that the training data be split into Input-Target pairs also called Prompt-Response pairs. In our case, the questions were used as Prompt and the answers were used as the response. However, before passing the training data to T5 model, the data was slightly modified to ensure good performance and efficient learning process. A prefix string was added to the the questions that were passed as the training prompt. The prefix defined was – "Answer the Math Question:". This was done due to the following reasons:

1. Contextualization: The prefix provides context to the model. Without a task-specific prefix, the model might struggle to understand what is expected from the given input. The prefix acts as a guide, leading the model to generate an output that is aligned with the desired task.

2. Improved Performance on Specific Tasks: The model can learn task-specific nuances and subtleties better when it is explicitly trained with task prefixes. This focused learning often leads to improved performance on those tasks.

3. Consistent Formatting: Prefixes ensure a consistent format in the input data, which is important for the model to process information correctly. This consistency is especially crucial when dealing with diverse datasets or multiple tasks within the same dataset. As T5 comes trained on multiple tasks, adding a prefix for our task helps the model learn and predict better as it allows a subtle differentiation from other tasks it is trained on like translation from English to German language.

Question	Answer	Gen Answer
Jimmy is at the candy store and buys 2 candy bars for $.75 each. He then buys 4 lollipops that cost $.25 each. He spent 1/6 of the money he earned from shoveling snow. If he charges $1.5 per driveway, how many driveways did he shovel?	He spent $1 on lollipops because 4 x .25 = «4*.25= 1» 1. He spent $2.5 in total because 1.5 + 1 « 1.5+1=2.5» 2.5. He earned $15 shoveling driveways because 2.5 / (1/6) = «2.5/(1/6) =15»15 He shoveled 10 driveways because 15 / 1.5= «15/1. 5=10» 10 #### 10	He spent $1.5 per driveway. He spent $.25 per lollipops. He spent $.25 per lollipops. He spent 1/6 of the money he earned from shoveling snow. #### 10

Table 3: Sample output generated by T5 model

Question	Answer	Gen Answer
Eddy's spider plant produces 2 baby plants 2 times a year. After 4 years, how many baby plants will the mother plant have produced?	The mother plant produces 2 baby plants 2 times a year so it produces 2*2 = «2*2=4»4 plants a year It produces 4 plants a year so after 4 years it will produce 4*4 = «4*4=16» 16 baby plants #### 16	Since 1 baby plant is produced twice a year, the mother plant will have produced 2 * 4 = «2*4=8» 8 baby plants in 4 years. In 4 years , she will have produced 2 * 8 = «2*8=16» 16 baby plants. #### 16

Table 4: Sample output generated by GPT-2 model

4. Efficient Fine-Tuning: When fine-tuning the model for specific tasks, prefixes make it easier to adapt the model to new tasks or variations of tasks.

The training data was then tokenized using the Autotokenizer imported from Huggingface's transformers. The T5 model was initially trained with a learning rate of 2e-5, a batch size of 16, and over 3 epochs. The training arguments specified for T5 reflect a balance between computational resources and the need for a robust model fine-tuning, with adjustments made for batch size and learning rate to optimize performance.

GPT-2 (Generative Pre-trained Transformer 2)

GPT-2, known for its strong language generation capabilities, is built on a transformer architecture with an emphasis on unsupervised learning. It is pretrained on WebText, a dataset gathered by OpenAI by scraping several internet websites whose links are found on highly rated Reddit threads. It can generate coherent and contextually relevant text passages, making it apt for producing step-by-step solutions to math word problems. The transformer architecture allows it to handle sequences effectively, which is crucial when dealing with the logical sequence of steps in math problems.

The fine-tuning involved a learning rate of 5e-5 and batch size of 4, tailored to balance the quality of output against the limitations of available computing power, ensuring the model's training is both effective and feasible. The model was trained on Google Colab GPUs for 3 epochs and the same split of 80% training and 20% test was used. As displayed in Table 4, the output from the fine tuned GPT-2 model was the best out of all three models, which is further discussed in the Results section.

Improving Performance

Our BLEU scores were initially disappointingly low, indicating a gap between the model-generated solutions and the reference solutions. This low score was a concern as it suggested that the models were not accurately capturing the nuances of mathematical problem-solving or aligning closely with the expected solutions.

We first implemented weight decay in our models, a common regularization technique. The rationale behind this was to address potential overfitting issues. Overfitting occurs when a model learns the training data too well, including its noise and outliers, which reduces its ability to generalize to new data. By penalizing large weights in the model's parameters, weight decay helps in regularizing the model. This means it discourages the model from becoming too complex and fitting too closely to the training data. A simpler model, in theory, should generalize better to new data, potentially improving BLEU scores as the solutions might align more closely with the varied ways problems can be correctly solved.

The second approach was to implement a scheduled learning rate. The learning rate is a crucial hyperparameter in training neural networks. It determines the size of the steps the model takes during optimization. If the learning rate is too high, the model might overshoot optimal solutions; if it's too low, the model might not converge or take too long to train. We designed the learning rate to increase linearly during the first 10% of the training steps, starting from zero. This gradual increase helps the model initially explore a broader solution space. After reaching this peak, the learning rate then begins to decrease, allowing the model to fine-tune and converge more precisely on optimal solutions as it approaches a loss value closer to zero. This strategy aims to strike a balance between exploration and exploitation. In the early stages of training, the model is encouraged to explore a wide range of solutions, which is crucial for understanding the diverse ways mathematical problems can be solved. As the training progresses, the focus shifts to refining and honing in on the most effective solutions. This approach is expected to lead to better model performance and, consequently, higher BLEU scores.

While applying these methods helped us improve the BLEU scores, the difference was not as much as we had hoped as the average improvement across the two models was by 0.005. The performance of our models is evaluated below and the results are discussed and analyzed.

Evaluation

To rigorously assess the performance of our models, we utilized three established metrics: BLEU (Bilingual Evaluation Understudy), BERTScore, and STS (Semantic Textual Similarity). The BLEU score provided a measure of the correspondence between the model-generated solutions and the actual answers, reflecting the quality of the text generation. BERTScore, leveraging the contextual embeddings from BERT, measured the semantic similarity between the generated and the true solutions. Finally, STS offered a fine-grained analysis of the cosine similarity between the semantic representation of generated and true answers on a sentence level, indicating how closely the models' outputs matched the expected solutions in terms of meaning and structure.

Model	BLEU	STS	F1
5-Gram	0.0032	0.19	0.73
T5	0.0051	0.73	0.82
GPT-2	0.03	0.81	0.88

Table 5: Calcluated Results

Results

Upon evaluation, the results revealed a clear distinction in performance among the models. The 5-Gram model, while notable for its computational swiftness, exhibited a BLEU score close to 0.0032, indicating a relatively poor match with the expected answers. Additionally, it achieved an STS cosine similarity score of 0.19, suggesting limited semantic alignment with the true answers, and it was unable to generate any correct math answers.

In stark contrast, the T5 model marked a considerable improvement, with a BLEU score of 0.005 and an STS cosine similarity of 0.73. It successfully generated nine correct math answers. Its BERTScore F-1 measure, combining precision and recall, stood at 0.82, signifying a robust capacity to produce relevant and precise solutions.

The GPT-2 model further enhanced these outcomes, reaching a BLEU score of 0.13 and an STS cosine similarity score of 0.80, reflecting a strong semantic resemblance to the target answers. The model provided eight correct math solutions, and its BERTScore F-1 was at an impressive 0.88, indicating its proficiency in capturing the essence of the math problems and producing accurate answers.

Analysis

These results collectively suggest that transformer-based models like T5 and GPT-2 are not only more adept at generating text that is semantically similar to actual solutions but also show promise in actual problem-solving capability, a significant step forward in applying NLP techniques to educational tools.

In our case, the BLEU score, a metric typically used for evaluating the quality of text translation, was applied to assess the solutions generated by our transformer language models for mathematical problems. The low score observed indicates that there is a discrepancy between the model-generated solutions and our chosen reference solutions.

Key Factors Contributing to the Low BLEU Score:

1. Variability in Mathematical Expression: Mathematical problems often have solutions that can be correctly expressed in multiple ways. The language model might produce an accurate and valid solution that is structurally or lexically different from the reference solution. Since BLEU emphasizes exact word and phrase matches, it penalizes these valid but differently worded responses, leading to a lower score.

2. Complexity of Reasoning Steps: Our model generates not just answers but also the reasoning steps leading to those answers. The complexity and variability in explaining these steps can significantly differ from the reference solutions, impacting the BLEU score. This is particularly relevant for more complex problems where multiple reasoning paths are possible.

3. Semantic Accuracy vs. Lexical Matching: The BLEU score does not adequately capture semantic accuracy — it focuses on the surface-level lexical similarity. Hence, the model's solutions might be semantically correct and logically sound but still score low on BLEU due to lexical differences.

However, higher STS and F1 scores indicate that the GPT-2 and T5 models were able to capture the essence of the solution in their generated answers in a lot of the cases. The capabilities demonstrated by transformer-based models like T5 and GPT-2 in solving mathematical problems mark a significant stride in applying NLP techniques to educational tools. Their ability to understand complex problem statements, apply logical reasoning, and adapt to various problem types positions them as invaluable assets in both educational and professional realms.

Potential Next Steps

We believe that this project can be extended and further developed. Firstly, one of the major shortcomings of our models was that they often didn't produce the final numeric answer. This can be resolved by adding a seperate output that just predicts the final numberic answer and a different output for the steps in natural language. Other models such as BERT could also be fine tuned for this task and a custom transformer model could be developed too that is primarily trained on a large corpus of mathematical information and then fine tuned on problem sets containing word problems. Combining different datasets in training could also help the model perform better.

Conclusion and Discussion

This project represents a significant leap forward in the intersection of advanced natural language processing and mathematical problem-solving. By integrating the sophisticated capabilities of transformer-based models with targeted fine-tuning in the realm of mathematical language, we have developed a tool that can potentially adds immense educational and practical value. Additionally, by comparing different models, we have shed a light on what works best for mathematical language modelling.

This tool is poised to revolutionize how students learn, how educators teach, and how professionals across various fields tackle complex mathematical challenges. It stands as a testament to the power of AI in bridging the gap between linguistic understanding and logical, quantitative reasoning, marking a pivotal moment in the ongoing evolution of AI applications in education and beyond. Our work lays the groundwork for further innovations in this field and opens up new possibilities for the application of AI in solving some of the most intricate problems faced by learners and professionals alike.

REFERENCES

1. I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112, 2014.

2. D. Huang, J. Liu, C.-Y. Lin, and J. Yin. Neural math word problem solver with reinforcement learning. In Proceedings of the 27th International Conference on Computational Linguistics, pages 213–223, 2018.

3. K. Chen, Q. Huang, H. Palangi, P. Smolensky, K. D. Forbus, and J. Gao. Mapping natural-language problems to formal-language solutions using structured

4. Cobbe, Karl, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo

Jun, Lukasz Kaiser, and Matthias Plappert. “Training Verifiers to Solve Math Word Problems.” arXiv, 2021. https://doi.org/10.48550/ARXIV.2110.14168.