NLP Group 1: Translating Math Problems

Paul Kelly, Carrie Magee, Jack McMorrow, Akshay Verma

Introduction

Language complexity gets tricky, especially when mixing math into word problems. Our group took on this challenge using transformers to simplify the complexities in math and languages

Math word problems, those short stories with real-world scenarios, need more than just basic math skills. You've got to understand the context, sentence structure, and word connections. Solving them means figuring out the problem, picking out the important info, and turning it into solvable math. Computers find this tricky due to ambiguity and understanding context.

Goals of the project

In our attempt to solve this problem, the multimodal capacities of transformers emerged as valuable assets. The goal of our project is to solve linguistic challenges with computational solutions, more specifically use the power of deep learning to convert word problems into solvable mathematical equations.

Dataset

In our study, we used two main datasets: MAWPS (A Math Word Problem Repository) for training and SVAMP (Simple Variation on Arithmetic Math Word Problems) for testing. The original research paper identified issues with widely used math word problem (MWP) benchmark datasets like ASDiv-A and MAWPS. Existing models performed well on these datasets, even when the “question” part was omitted during testing, indicating a reliance on cues unrelated to the actual math problem.

To address this, the researchers created the “SVAMP” dataset as a testing framework to evaluate a model’s proficiency in various aspects of mathematical word problem solving. SVAMP assesses sensitivity to questions, reasoning ability, and invariance to structural alterations. For example, it challenges models

with variations like changing the direction of interactions or altering the sequence of events within a problem.

PROBLEM: Text: Jack had 8 pens and Mary had 5 pens. Jack gave 3 pens to Mary. How many pens does Jack have now? Equation: 8 - 3 = 5
QUESTION SENSITIVITY VARIATION: Text: Jack had 8 pens and Mary had 5 pens. Jack gave 3 pens to Mary. How many pens does Mary have now? Equation: 5 + 3 = 8
REASONING ABILITY VARIATION: Text: Jack had 8 pens and Mary had 5 pens. Mary gave 3 pens to Jack. How many pens does Jack have now? Equation: 8 + 3 = 11
STRUCTURAL INVARIANCE VARIATION: Text: Jack gave 3 pens to Mary. If Jack had 8 pens and Mary had 5 pens initially, how many pens does Jack have now? Equation: 8 - 3 = 5

Table 1: Example of a Math Word Problem along with the types of variations that we make to create SVAMP.

MAWPS:

- MAWPS was developed by researchers at Google and Microsoft to be used to test various NLP models curated to solve math word problems.
- Inconsistencies in the performance of models on this dataset because when the question component of the problem was removed unring test the model still preformed well.
- Suggests potential reliance on cues unrelated to the actual mathematical concepts

SVAMP:

- Researchers developed SVAMP in response to the shortcomings of the MAWPS dataset.
- This testing dataset used to assess models' proficiency in various aspects of math word problem solving by including MWP that vary in question sensitivity, reasoning ability, and structural invariance

---

Aspect of Language Structure to Consider:

- Question Sensitivity: variations in SVAMP check if the model's answer depends on the structure of the question.
- Reasoning Ability: variations ensure the model has learned to correctly determine a change in reasoning arising from changes in the problem text.
- Structural Invariance: ensures that the model remains invariant to superficial changes in the problem text (i.e., changes that do not alter the answer or reasoning)

Training and Testing Dataset

We trained our model on an augmented version of MAWPS with approximately 60,000 rows and used SVAMP with 1000 math word problems for testing. SVAMP includes scenarios focusing on subtraction, addition, division, and multiplication. The dataset provides information about the question, numbers,

equations, and answers. The 'Numbers' column includes relevant numerical values for each problem, serving as inputs during data preprocessing. The 'Equation' column represents the target variable, aiding the evaluation of the model's ability to translate word problems accurately into a numeric format.

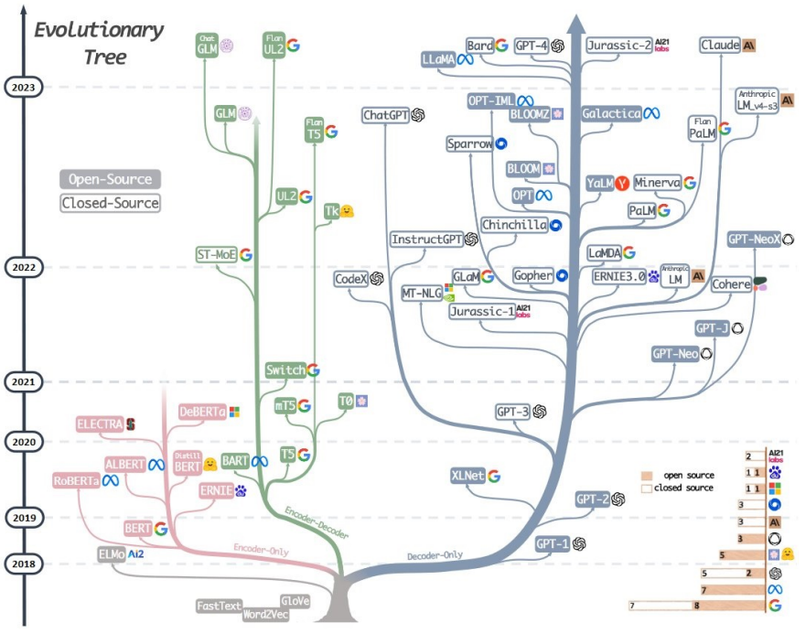
NLP Group 1: Translating Math Problems

Paul Kelly, Carrie Magee, Jack McMorrow, Akshay Verma

NLP Model Architecture

Original Attempts

We explored various deep learning approaches in order to achieve our goal of translating math word problems into a numeric output. The first approach attempted to use a GPT-2 type transformer due to its flexibility in various NLP tasks like language translation, summarization, and generation. Though the model showed extremely low loss after training, it ultimately was unable to translate math word problems into numeric equations thus, leaving the team to explore other sequence-to-sequence focused techniques.

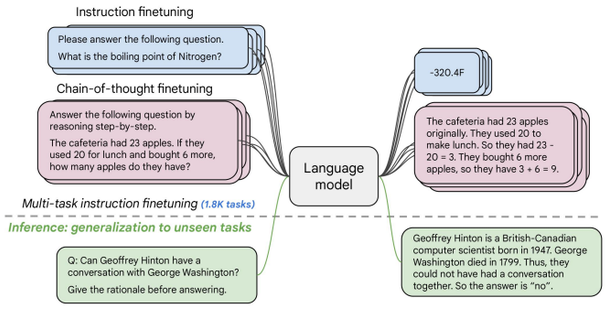


GPT Model: The GPT-2 type transformer follows the transformer architecture, known for its attention mechanisms. It excels in capturing contextual information across sequences. Cross-entropy loss was employed, commonly used in language modeling tasks, with the expectation that it would guide the model to generate accurate numeric equations for math word problems.Despite achieving low training loss, the model struggled to make accurate predictions, leading the team to reevaluate the chosen approach.

BERT Model: BERT, or

Bidirectional Encoder Representations from Transformers, is a transformer-based model designed for bidirectional context understanding. It is particularly effective in capturing dependencies in both directions.
: Similar to GPT-2, cross-entropy loss was applied, aiming to guide the model in understanding the sequential information in math word problems. BERT, too, faced difficulties in accurately translating math word problems into numeric equations, prompting a reassessment of the chosen architecture.

Flan T5 Model



Flan T5 Base, an extension of Google's T5 architecture, specializes in language-related tasks, with a particular focus on mathematical language understanding. Boasting 245 million parameters, this variant offers versatility for natural language processing (NLP) tasks, treating them uniformly as text-to-text problems. Through fine-tuning on the MAWPS augmented dataset, Flan T5 Base becomes adept at converting mathematical word problems to equations, capitalizing on the models adaptability inherited from the T5 architecture.

The model's efficacy in addressing the nuances of mathematical language can be attributed to its robustness and adaptability. Like its predecessor T5, Flan T5 Base utilizes standard language modeling loss, such as cross-entropy loss, during training. This ensures the generation of accurate equations that faithfully represent the mathematical relationships described in word problems. The fine-tuning process is crucial, involving experimentation with hyperparameters to attain an optimal configuration for the specific task at hand, thereby enhancing the model's performance in converting MWPs to equations

In summary, Flan T5 Base, with its foundation in the T5 architecture, proves to be a powerful tool for mathematical language understanding. Through fine-tuning on the MAWPS augmented dataset and a meticulous optimization process,

the model excels in converting complex mathematical word problems into accurate equations, showcasing its adaptability and robustness in handling diverse NLP challenges.

NLP Group 1: Translating Math Problems

Paul Kelly, Carrie Magee, Jack McMorrow, Akshay Verma

Experimental Design

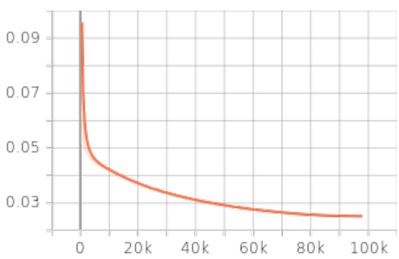
Before discovering our augmented dataset, we initially worked with a dataset that had distinct Numbers and Questions columns. To facilitate model training, we developed a function that utilized regular expressions to replace number placeholders within the questions. Specifically, for the T5-small model, we implemented a class that created customized training and validation datasets. In this setup, the processed Question (with real numbers) and Equation fields served as inputs, while the Answer field served as the output. Our Seq2Seq trainer was then instantiated with the specified training arguments.

In the context of equation generation models, we transitioned to using the Flan T5 Base model after experimenting with T5-small. The setup for Flan T5 Base mirrors that of T5-small, maintaining consistency in the approach. This transition allowed us to leverage the enhanced capabilities of Flan T5 Base for our specific task of converting mathematical word problems to equations, ensuring a seamless integration into our existing framework.

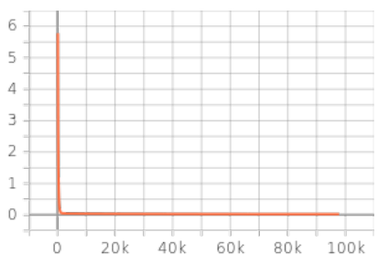
Hyperparameters

We employed Optuna for hyperparameter tuning, conducting multiple experiments to determine the optimal metrics for model evaluation. Initially, we explored minimizing loss and optimizing for exact matches and token-level accuracy between predicted and true answers in the validation set. However, tuning for token-level accuracy and exact answer match proved unsuccessful. The resulting model failed to produce correct validation predictions.

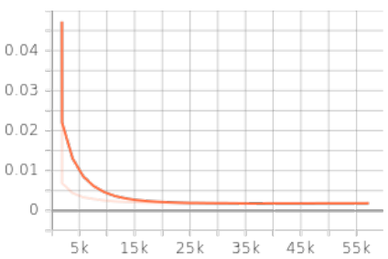
Subsequently, we focused on minimizing loss, and our tuner selected the following hyperparameter values: batch size - 64, epochs - 47, optimizer - Adam, and learning rate - 1e-4, resulting in a validation loss of 0.04. Unfortunately, the model produced with these parameters did not yield satisfactory results. After further experimentation, we settled on a batch size of 16, 200 epochs, optimizer Adam, and a learning rate of 1e-5, achieving an 80% correct prediction rate on our validation set.



Eval Loss for Numerical Answers



Train Loss for Numerical Answers



Eval Loss for Equations

NLP Group 1: Translating Math Problems

Paul Kelly, Carrie Magee, Jack McMorrow, Akshay Verma

Results

Metrics: Accuracy

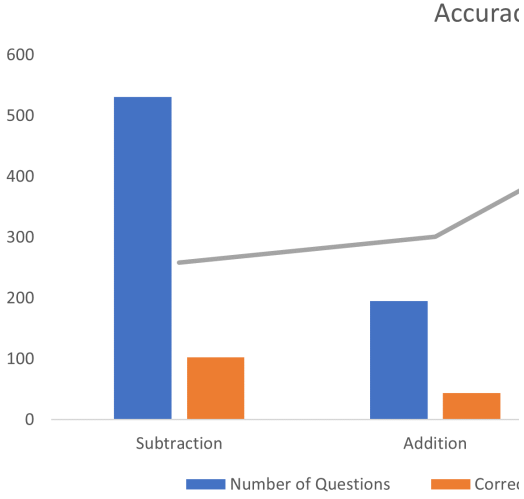
We use accuracy as our primary metric for assessing mathematical output, employing the sympify function to verify equation equivalence. Additionally, we incorporate ROUGE scores to gauge the quality of generated equations by measuring overlap and similarity with reference equations.

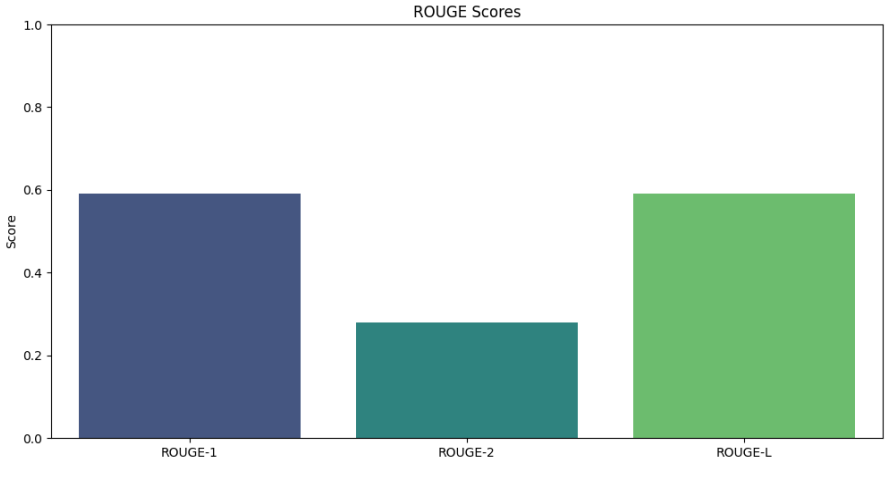
Our test questions cover four arithmetic types: Subtraction (531), Addition (195), Common-Division (166), and Multiplication (108). This categorization enables a thorough assessment of our models' problem-solving skills across diverse mathematical contexts.

Metrics: ROUGE

The ROUGE scores (ROUGE-1: 0.605, ROUGE-2: 0.287, ROUGE-L: 0.605) assess the Language Model's (LLM) linguistic performance, measuring overlap and similarity with reference equations. These scores highlight the model's proficiency in reproducing unigrams, bigrams, and maintaining linguistic coherence

A notable difference between ROUGE-2 and ROUGE-1 scores provides insights into the model's language generation capabilities, suggesting challenges in reproducing consecutive word sequences (bigrams) when ROUGE-2 is significantly lower than ROUGE-1.

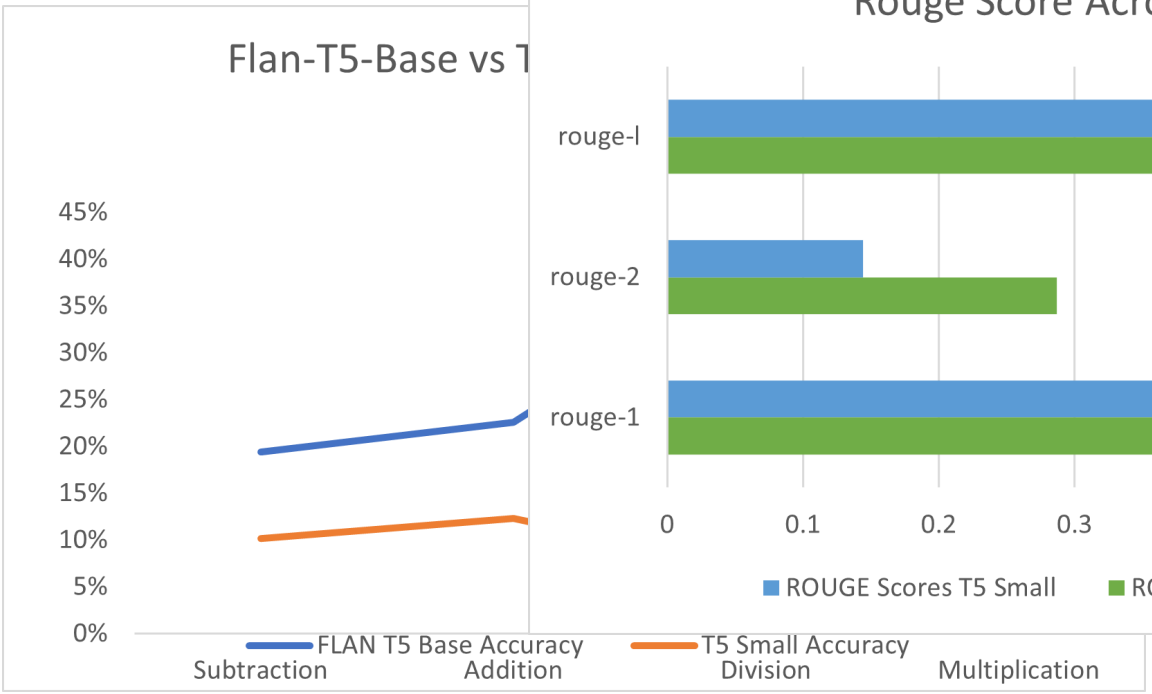




Flan T5 Base vs T5 small

Highlighting the crucial impact of model size and capacity on achieving higher accuracy rates, these results underscore Flan T5 Base's superior performance.

Flan T5 Base consistently outperforms T5 Small across arithmetic operations, with notable differences in Subtraction (19% vs. 10%), Addition (23% vs. 12%), Division (40% vs. 7%), and Multiplication (21% vs. 5%). In this comparison, Flan T5 Base achieves an overall accuracy of 23.6%, significantly outpacing T5 Small, which stands at 9.4%.



Despite the significant variance in overall accuracy between Flan T5 Base and T5 Small, a more nuanced perspective emerges when considering ROUGE scores.

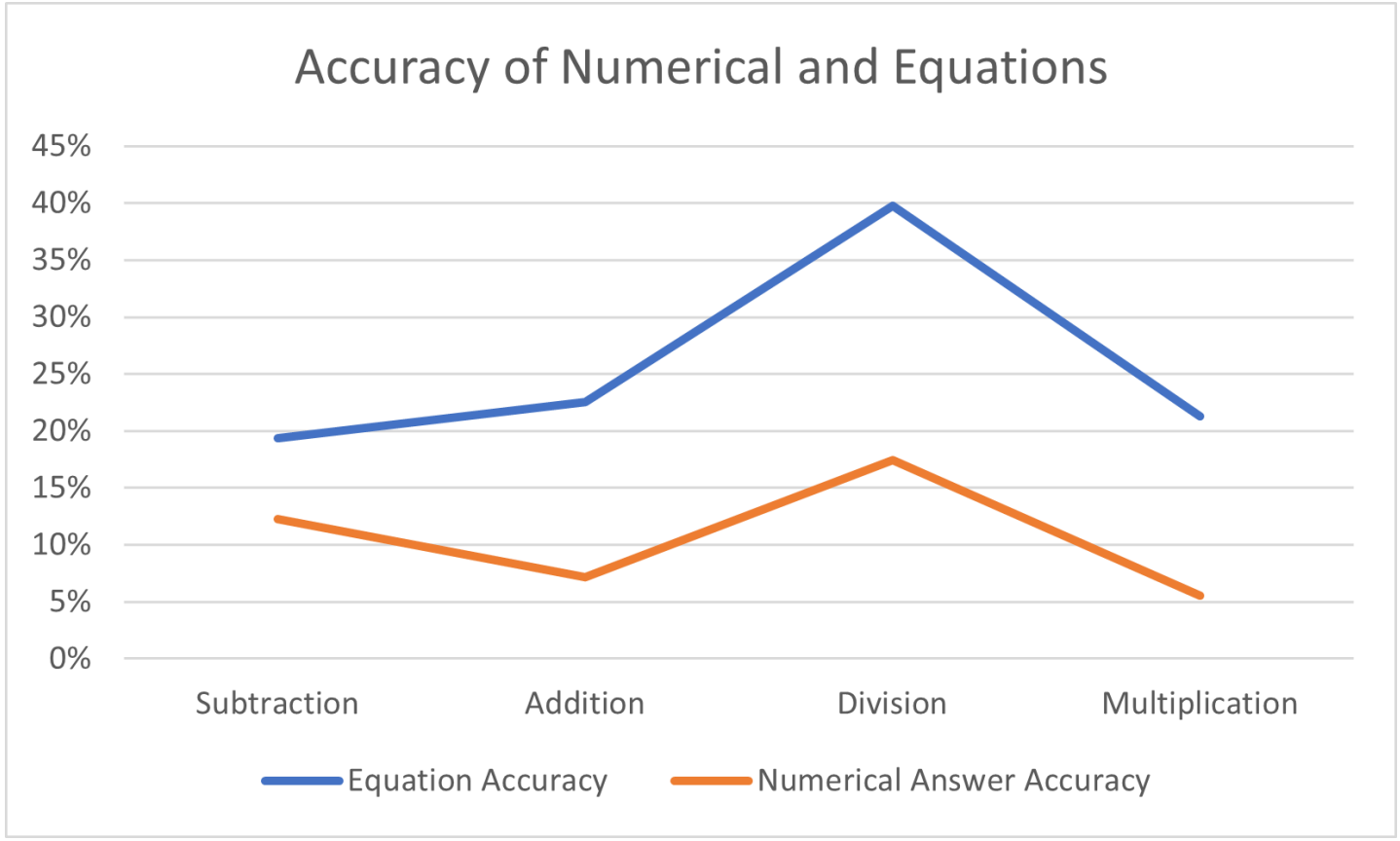
Flan T5 Base attains higher scores in both ROUGE-1 (0.605 vs. 0.537) and ROUGE-L (0.605 vs. 0.537), highlighting its superiority in unigram overlap and linguistic coherence.

Nevertheless, the substantial decline in ROUGE-2 scores (0.287 vs. 0.144) indicates challenges for both models in accurately reproducing consecutive word sequences.

The noticeable differences in performance between Flan T5 Base and T5 Small can be attributed to various factors, with model size and capacity being the primary influence. As a larger model, Flan T5 Base has a higher parameter count and inherent complexity, enabling it to capture and generalize more intricate patterns within the data.

Accuracy difference between numerical answers generation and equation generation

The performance gap between our equation-answering model and a numerical-focused counterpart reveals an intriguing trend. While the equation-answering model demonstrates commendable accuracy, the numerical model faces challenges, with Subtraction at 12%, Addition at 7%, Common-Division at 17%, and Multiplication at 6%. This suggests potential complexities in the model's numerical reasoning, prompting further examination for future enhancements to optimize accuracy in providing correct numerical solutions.



Limitation

Despite the promising features of the Flan T5 Base model, it's essential to acknowledge its limitations. The model's accuracy falls short when faced with more complex mathematical word problems, demonstrating a noticeable struggle in handling higher-level mathematical concepts. While adept at basic arithmetic and straightforward problem-solving, its performance tends to plateau, making it less reliable for intricate or advanced mathematical scenarios. This limitation underscores the importance of continued research and development to enhance the model's capabilities and extend its applicability to a broader range of mathematical complexities. Acknowledging these constraints provides a transparent understanding of the model's current limitations and serves as a foundation for future improvements and advancements in mathematical language understanding models.