Carrie Magee, Akshay Verma, Jack McMorrow, Paul Kelly
DATS 6312 - Natural Language Processing for Data Science
12/11/2023

Using Deep Learning to Bridge Language and Numbers

Introduction

Inherent ambiguity serves as the epitome of complexity in human language, and the introduction of math and language, in the context of word problems, further accentuates this intricacy. Nevertheless, our group aimed to confront this challenge by leveraging the power of transformers to unravel the intricacies woven into the phenomena of mathematical linguistics.

Math word problems, comprising short natural language narratives, present exercises that often involve real-world scenarios. These scenarios require individuals to apply their skills to interpret and solve problems, requiring not only a grasp of basic math but also an understanding of context, sentence structure, and word dependencies. The process of solving math problems entails understanding the given problem, prioritizing relevant information, and converting the written language problem into solvable mathematical expressions. Considering the intricate steps involved in solving word problems, it is evident that they pose a distinctive challenge to computers, primarily due to the necessity for ambiguity resolution and contextual understanding.

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 Description

Our study leveraged two primary datasets, MAWPS (A Math Word Problem Repository) and SVAMP (Simple Variation on Arithmetic Math Word Problems), for training and testing, respectively. The research paper that inspired the current project identified shortcomings in widely used math word problem (MWP) benchmark datasets like ASDiv-A (Academia Sinica Diverse) and MAWPS (Patel et al., 2021). Notably, existing models performed well on these datasets even when the “question” component was omitted during testing, indicating a reliance on cues unrelated to the actual mathematical problem.

In order to facilitate a more comprehensive evaluation of automatic MWP solvers, the researchers created the “SVAMP” dataset, as a testing framework, to assess a model’s proficiency in various aspects of mathematical word problem solving. SVAMP assesses a model’s proficiency in various aspects of mathematical word problem solving by including math word problems that vary in question sensitivity,reasoning ability, and structural alterations. An example of question sensitivity variation can be observed in the following mathematical word problem: *“Jack had 8 pens, and Mary had 5 pens. Jack gave 3 pens to Mary. How many pens does Mary have now?”* By including examples like this in the testing set, we are better able to assess our model’s ability to integrate information from different parts of a problem. SVAMP also addresses reasoning ability like in the word problem *“Jack had 8 pens and Mary had 5 pens. Mary gave 3 pens to Jack. How many pens does Jack have now?”* This example challenges the model’s reasoning skills by introducing a problem where the usual direction of the interaction is reversed. A model with significant reasoning abilities should be able to adapt to such variations.

Examining the last variation in the SVAMP dataset, structural invariance relates to how well a model can maintain accuracy in its solutions despite changes to the sequence of events within a problem. For example, the scenario *“Jack gave 3 pens to Mary. If Jack had 8 pens and Mary had 5 pens initially, how many pens does Jack have now?”* Although the information is the same in the prior examples, the structural rearrangement challenges the model to adapt to a different presentation of the problem.

Following the framework of the original research, we employed MAWPS as our training dataset. While the initial MAWPS dataset comprised 1921 rows, we enhanced our model’s robustness by utilizing an augmented version with approximately 60,000 rows (Mathbot, 2023). The MAWPS dataset contains a column for question and equations in the format of X = number operation number (e.g., X = 7.0 - 5.0). .

For our testing dataset, SVAMP consists of 1000 math word problems featuring 531 subtraction, 195 addition, 166 division, and 108 multiplication focused scenarios. The dataset contained columns for question, numbers, equations, and answers. The “Question” column contains the natural language representation of the mathematical world problem. This column serves as the foundation for assessing the model’s ability to understand the context of the given problem.

The “Numbers" column includes the numerical values relevant to each problem, which later serve as inputs during data preprocessing. These numbers are systematically substituted into the placeholders (number0, number1), providing the model with the quantitative information required for solving the mathematical word problems.

Significantly, the “Equation” column represents an equation derived from the question and body problem. It includes the operation sign (e.g., “-”, “+”) and the specific numerical values involved in the problem, such as number0, number1, and so on. This column functions as the target variable in the dataset and facilitates the evaluation of the model’s ability to accurately translate the word problem into a numeric format.

Overall, the SVAMP dataset presents a new challenge to the limitations of pre-existing MWP datasets by including variations of linguistic patterns in order to fully assess a model’s proficiency across various aspects of mathematical problem-solving.

Description of NLP Model

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. 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, 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.

Despite achieving low loss during training with GPT-2 and BERT, the models struggled with incorrect predictions when tasked with translating math word problems. This discrepancy between low loss and suboptimal accuracy highlighted a potential limitation in the chosen architectures. The intricate nature of mathematical language and the need for precise numeric representations demanded a different approach.

The Flan T5 model, an extension of the T5 architecture by Google, harnesses the capabilities of transformers, attention-based models known for capturing contextual information effectively. With a focus on language-related tasks, including mathematical language understanding, Flan T5 emerges as an ideal choice for processing your MAWPS augmented data. In our implementation, we specifically utilize the Flan T5 Base variant, equipped with 245 million parameters.

Designed as a variant of the T5 model, Flan T5 Base is finely tuned for various natural language processing (NLP) tasks, treating them uniformly as text-to-text problems. This approach provides a unified framework applicable to a broad spectrum of applications, ranging from text summarization to translation and question answering.

Flan T5 Base's effectiveness for our problem of converting mathematical word problems (MWPs) to equations can be attributed to its robustness and adaptability. As a variant of T5, Flan T5 inherits the versatility of the original model, making it well-suited for handling the nuanced language in MWPs. The fine-tuning on the MAWPS augmented data ensures that the model is specifically tailored to the intricacies of mathematical language.

Similar to T5, Flan T5 Base typically employs a standard language modeling loss, such as cross-entropy loss, during training. The fine-tuning process guides the model to minimize this loss, ensuring that it generates equations that accurately represent the mathematical relationships described in the word problems

In pursuit of our objectives, we have fine-tuned Flan T5 Base on the MAWPS augmented dataset. This fine-tuning process involves experimenting with hyperparameters to achieve an optimal configuration for the specific task at hand, enhancing the model's performance in converting MWPs to equations.

Experimental Setup

Before we located our augmented dataset, we utilized the set detailed above, which included separate Numbers and Questions columns. We wrote a function that switched out number placeholders in the questions using a regular expression prior to training our models.

Then, for the T5-small model, we wrote a class to create custom training and validation datasets where the processed Question (including real numbers) and Equation fields served as inputs, and Answer as output. We then instantiated our Seq2Seq trainer with the training arguments detailed below.For the equation generation model, we used a Flan T5 Base- model after experimenting with T-5 small, the setup of the Flan T5 -Base is the same as that of T5-small.

For hyperparameter tuning, we used Optuna. We conducted several experiments as we narrowed down metrics upon which to base our model evaluation - first, tuning to minimize loss, but also exploring optimizing for exact matches between predicted output and true answer in the validation set; and, finally, for token-level accuracy (i.e. number of matching tokens between the prediction and true answer). The hyperparameters we tuned were: number of training epochs (between 1 and 100); learning rate (between 1e-3 and 1e-6); batch size (8, 16, 32, and 64); and optimizer (Adam, SGD, RMSprop, AdamW, and Adadelta). Tuning for token-level accuracy and exact answer match proved unsuccessful - the resulting model produced no correct validation predictions (and sometimes no predictions at all). For this reason, we focused on minimizing loss. Our tuner selected the following hyperparameter values: batch size - 64; epochs - 47; optimizer - Adam; learning rate - 1e-4, for a validation loss of 0.04. Ultimately, however, the model produced with these parameters did not produce satisfactory results. After further experimentation, we settled upon a batch size of 16, 200 epochs, optimizer Adam, and learning rate of 1e-5, which produced correct predictions from our validation set at a rate of 80%. The evaluation loss decreased progressively across epochs, as expected.

Figure 1. T5 Network Structure

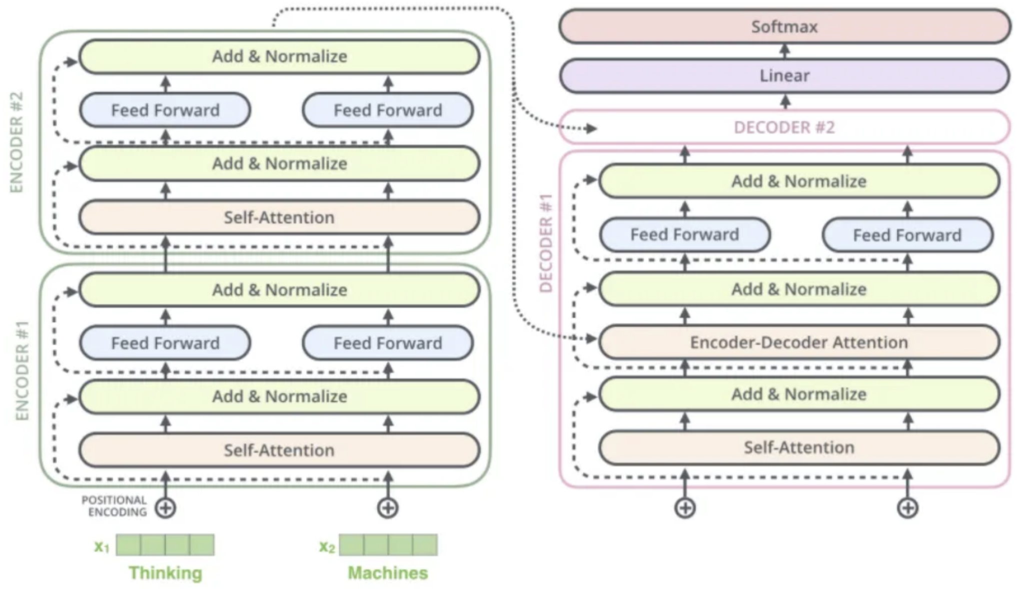
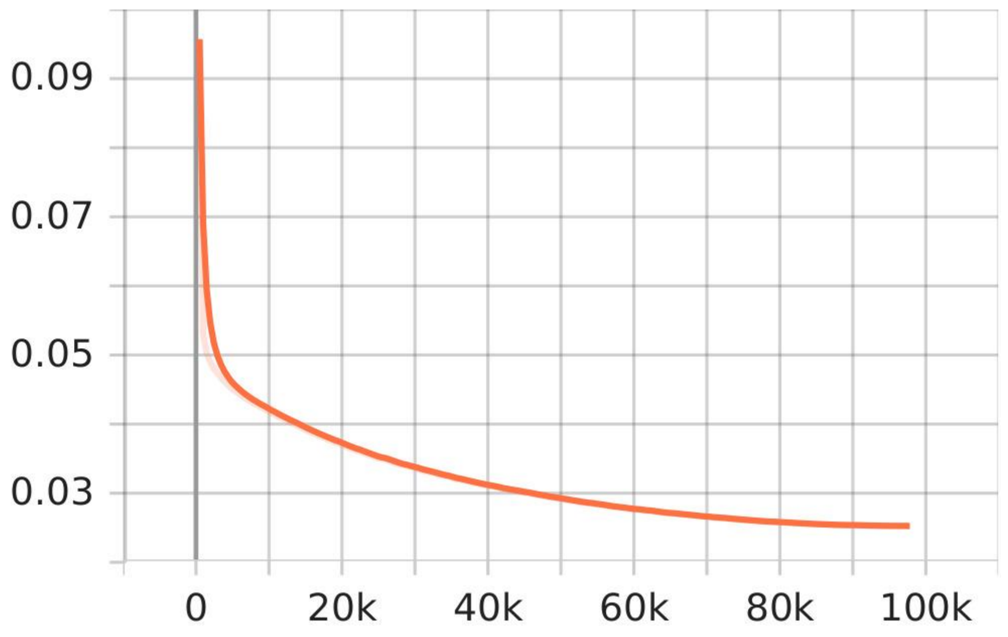


Figure 2. Evaluation Loss Across Epochs



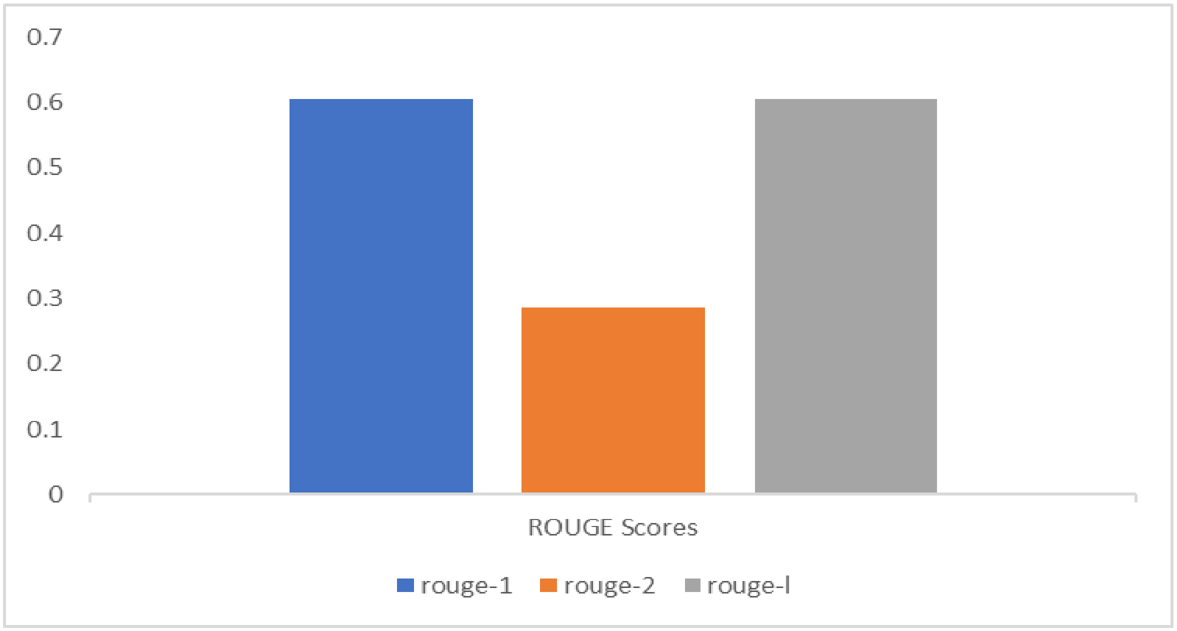
Results

The selection of metrics is essential for a nuanced assessment of our model. Our key metric is accuracy as we are dealing with mathematical outputs. For equation generation, we use the sympify function to check whether the predicted equation is equivalent to the actual prediction. In addition to accuracy, we incorporate ROUGE scores as supplementary metrics for checking the quality of generated equations. ROUGE scores measure the overlap and similarity between the model generated and reference equations by comparing the number of matching n-grams. Here we report ROUGE-1, which analyzes matching unigrams, ROUGE-2, which analyzes matching bigrams. ROUGE-L, on the other hand, matches the Longest Common Subsequence, which is the longest subsequence that occurs in both the generated output and the reference equation.

Our test questions span four distinct types based on arithmetic operations: Subtraction (531 questions), Addition (195 questions), Common-Division (166 questions), and Multiplication (108 questions). This categorization allows for a comprehensive evaluation of our Models' problem-solving capabilities across varied mathematical contexts. The model correctly answers 19.4% (103 out of 531) of Subtraction questions, 22.6% (44 out of 195) of Addition questions, 39.8% (66 out of 166) of Common-Division questions, and 21.3% (23 out of 108) of Multiplication questions.

The ROUGE scores, comprising ROUGE-1 (0.605), ROUGE-2 (0.287), and ROUGE-L (0.605), encapsulate the Large Language Model's (LLM) linguistic performance by quantifying the overlap and similarity with reference equations. These scores underscore the model's proficiency in reproducing unigrams, bigrams, and maintaining linguistic coherence, respectively. A significant discrepancy between ROUGE-2 and ROUGE-1 scores can offer valuable insights into the language generation capabilities of the model. When ROUGE-2 is notably lower than ROUGE-1, it suggests that the model faces challenges in reproducing consecutive word sequences (bigrams) from the reference equations.

Figure 3. Bar Chart of ROUGE Scores for Flan-T5-Base



The comparison between Flan T5 Base and T5 Small reveals distinct performance differences. Flan T5 Base boasts an overall accuracy of 23.6%, significantly surpassing T5 Small at 9.4%. Across arithmetic operations, Flan T5 Base consistently outperforms T5 Small. Notable differences include Subtraction (19% vs. 10%), Addition (23% vs. 12%), Division (40% vs. 7%), and Multiplication (21% vs. 5%). These results underscore Flan T5 Base's superiority, highlighting the pivotal role of model size and capacity in achieving higher accuracy rates.

Figure 4. Performance Differences between Flan-T5-Base and T-5 Small

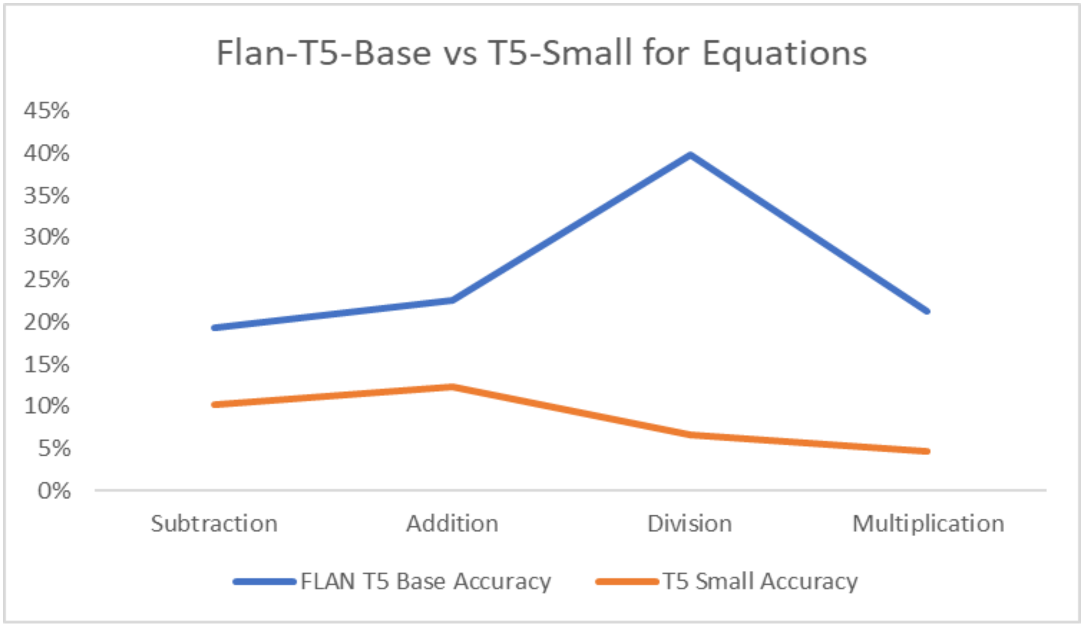
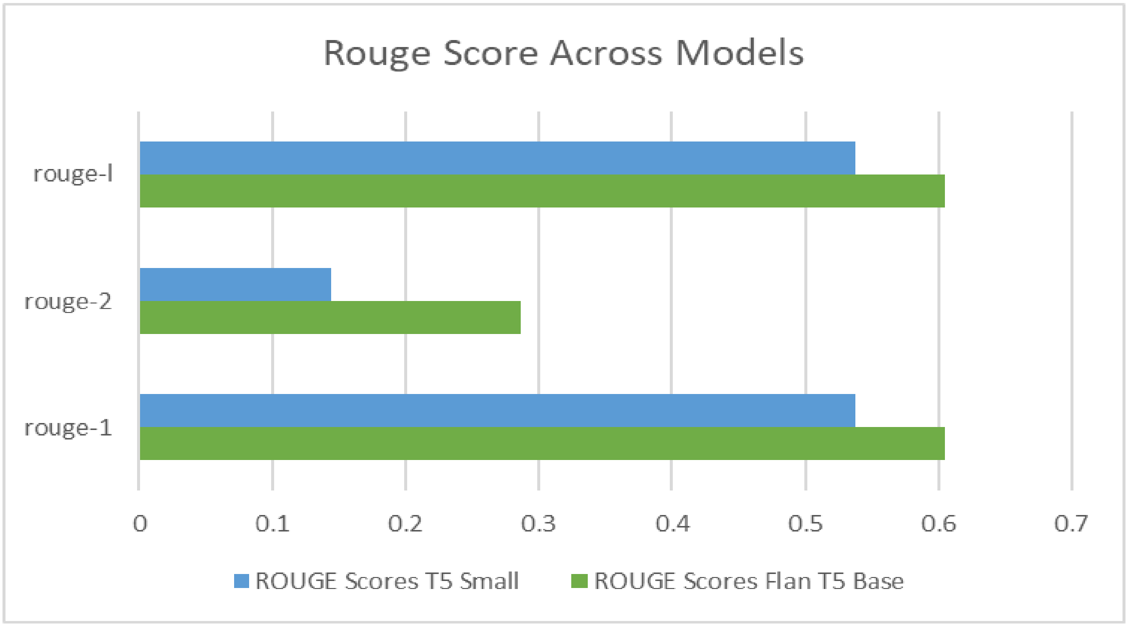


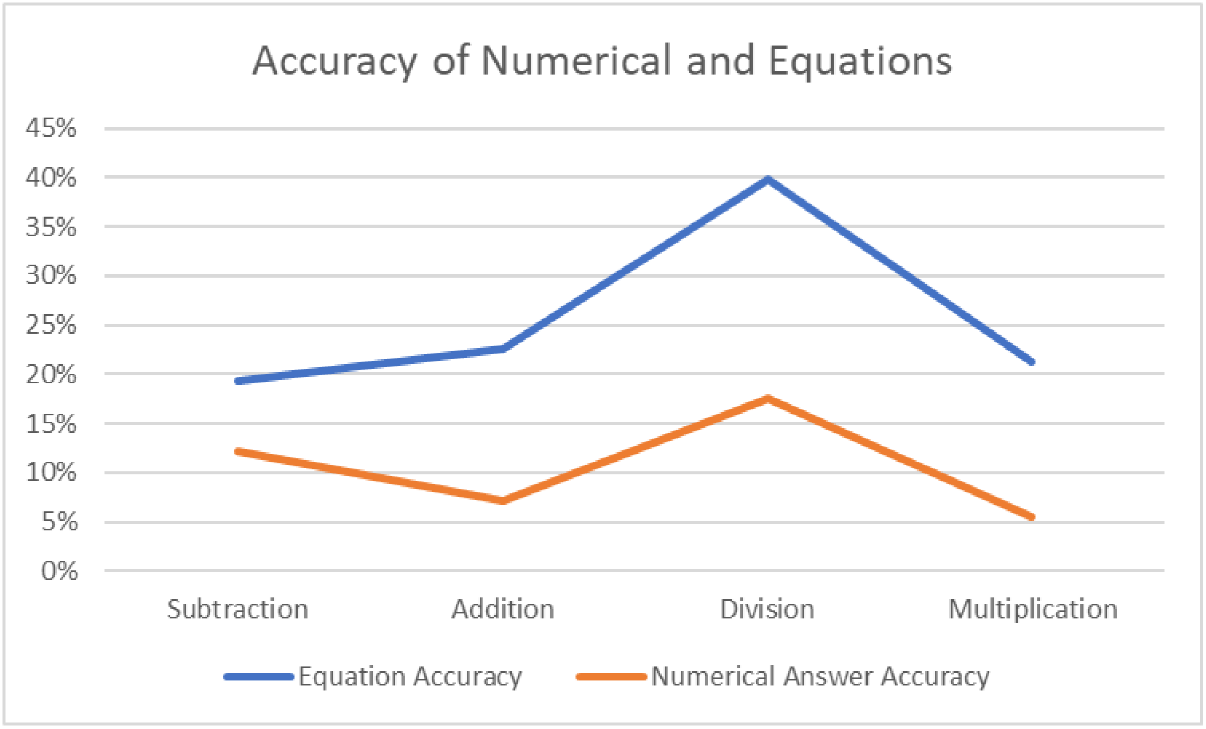
Figure 5. ROUGE Scores of Flan-T5-Base and T-5 Small



The discernible disparities in performance between Flan T5 Base and T5 Small can be attributed to several factors, with model size and capacity being the primary influencer. Flan T5 Base, being a larger model, possesses a greater parameter count and inherent complexity, allowing it to capture and generalize more intricate patterns within the data.

While the overall accuracy between Flan T5 Base and T5 Small displays a notable difference, the ROUGE scores reveal a more nuanced comparison. Flan T5 Base achieves higher scores across ROUGE-1 (0.605 vs. 0.537) and ROUGE-L (0.605 vs. 0.537), indicating its superiority in unigram overlap and linguistic coherence. However, the substantial drop in ROUGE-2 scores (0.287 vs. 0.144) suggests challenges for both models in reproducing consecutive word sequences accurately.

Figure 6. Equation vs. Numeric Answer Accuracy



The disparity in performance between our equation-answering model and a model exclusively handling numerical responses reveals an intriguing trend. While the equation-answering model displays commendable accuracy, the numerical-focused model encounters challenges, with Subtraction at 12%, Addition at 7%, Common-Division at 17%, and Multiplication at 6%. This distribution of correct answers highlights a discrepancy, indicating potential complexities in the model's ability to precisely compute and predict numerical outcomes. The diminished accuracy in numerical responses compared to the equation-answering counterpart prompts a closer examination of the model's numerical reasoning capabilities, guiding potential future enhancements to optimize its accuracy in providing correct numerical solutions.

Conclusion

Overall, the results of this project emphasize the robustness and power of transformers and LLMs in their ability to translate word constructed math problems into their underlying equations. After finding sufficient data and experimenting with different models, we were able to utilize two different T5 models in order to achieve a desired result.The Flan T5 base model performed better across the different ROUGE score, which measure the similarity between the desired target and the output of the transformer, as well as the overall accuracy of the results, which stood at 23.6%. The T5 small model had consistent accuracy across different operations, while the Flan T5 model performed the best on division operations.

Despite the satisfactory results, an accuracy of just twenty-four percent can definitely be improved upon. While the Flan T5 model has impressive architecture and has been proved to perform well on many different tasks, the final model we trained tends to fall short for more complex mathematical problems. The model performance tends to plateau, signifying an underlying limitation in its performance. In order for the model to improve, the capabilities of the model would need to be enhanced for our accuracy to increase. We could also explore other types of transformers and assess their performance for a problem like this. Regardless, the Flan T5 model's ability to assess these mathematical problems was impressive and shows the powers that Natural Language Processing has when it comes to translating MWPs into equations.

Work Cited

Doe, P. (2021). Rouge Your NLP Results. Medium. https://medium.com/@priyankads/rouge-your-nlp-results-b2feba61053a

Koncel-Kedziorski, R., Roy, S., Amini, A., Kushman, N., & Hajishirzi, H. (2016, June). MAWPS: A math word problem repository. In Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies (pp. 1152-1157).

Hugging Face. (2023). T5-small: Text-To-Text Transfer Transformer. Hugging Face. https://huggingface.co/t5-small

Hugging Face. (2023). T5-Flan-Base: Text-To-Text Transfer Transformer. Hugging Face. https://huggingface.co/google/flan-t5-base

Jacob, John. (2023). What is FLAN-T5? Exemplary AI Blog. https://exemplary.ai/blog/flan-t5

Patel, A. (2022). SVAMP: Structural Variant Annotation, Mapping, and Prediction. GitHub. https://github.com/arkilpatel/SVAMP

Patel, A., Bhattamishra, S., & Goyal, N. (2021). Are NLP models really able to solve simple math word problems?. arXiv preprint arXiv:2103.07191.

MathBot. (2023). MAWPS_Augmented.pkl. GitHub. https://github.com/Starscream-11813/MathBot/blob/main/MAWPS_Augmented.pkl

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21(140), 1-67.