Paul Kelly
DATS 6312 - Natural Language Processing for Data Science
Final Project - Individual Report
The George Washington University
12/11/2023

Introduction

Our group project aimed to use transformers to both translate simple math word problems into equations, but also generate answers to those problems not via mathematical calculations, but through machine learning. The shared work was as follows: each member initially explored the problem themselves, producing basic code; then, we worked together on the code before settling on two models we would refine together. Then, we each shared augmented versions of the datasets before finally settling on Akshay's Flan T5 Base model for producing equations and my T5 Small model for answers. We worked collaboratively on ingesting our models to Hugging Face, and Akshay and Jack worked on coding our Streamlit presentation. Carrie took the lead on our final written report, to which we all contributed.

Individual Work

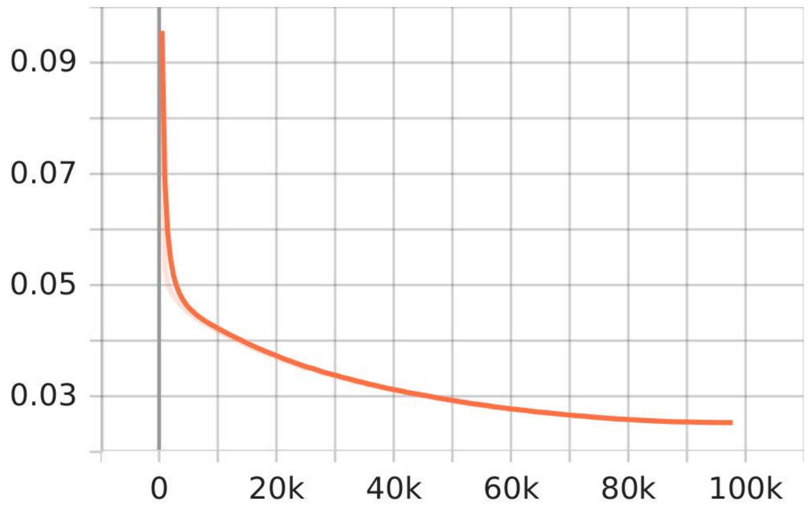
My individual work included setting up and maintaining our group Github repository, coding our hyperparameter tuner, our successful T5 Small Seq2Seq model, and producing Tensorboard and ONNX outputs.

Individual Work in Detail

In detail, my T5 Small code performed the following tasks: dataset preparation - data was split into train and validation sets, and custom datasets were constructed using a custom class (input sequences were questions and equations, target sequences were answers); pre-processing - I wrote a function to replace number placeholders in the Question column with provided correct numbers in the Numbers column in train.csv to facilitate next steps; tokenization - data was tokenized with the t5 tokenizer; training configuration - here I set important parameters like batch size and number of epochs; training and evaluation - upon instantiating the Seq2SeqTrainer, I then defined the model, training arguments noted above, and, importantly, selected the Adam optimizer and custom train and val datasets; tensorboard logging to produce visualizations; then printing results from the validation set once training was complete to determine if model was performing correctly and producing numeric results; and finally, providing a sample question for the model to answer.

Beyond this specific script, I also wrote hyperparameter tuning code to tune number of epochs, optimizers, batch size, and learning rate. For hyperparameter tuning, I used Optuna. I 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, I focused on minimizing loss. The 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, I settled upon a batch size of 16, 200 epochs, optimizer Adam, and learning rate of 1e-5, which produced correct predictions from the validation set at a rate of 80%.

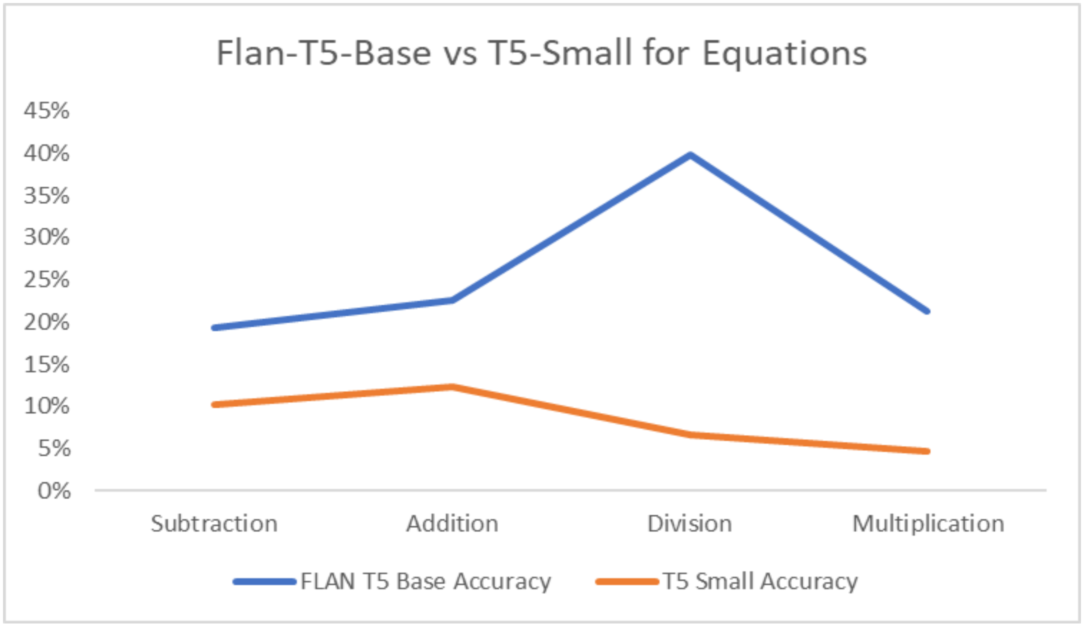
Figure 1: Evaluation Loss Across Epochs



Results

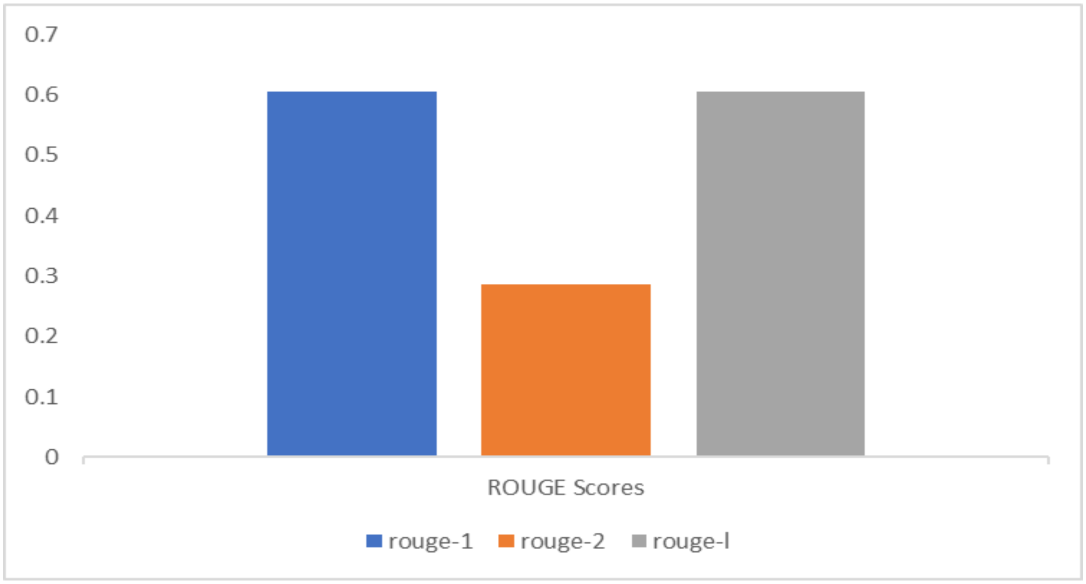
Accuracy was the primary evaluation metric for the mathematical outputs. Equations were assessed using the sympify function, with additional ROUGE scores measuring overlap and similarity. Our test questions spanned four arithmetic types, with the model achieving varied accuracy: Subtraction (19.4%), Addition (22.6%), Common-Division (39.8%), and Multiplication (21.3%).

Figure 2: Performance Differences between T5 Base and T5 Small



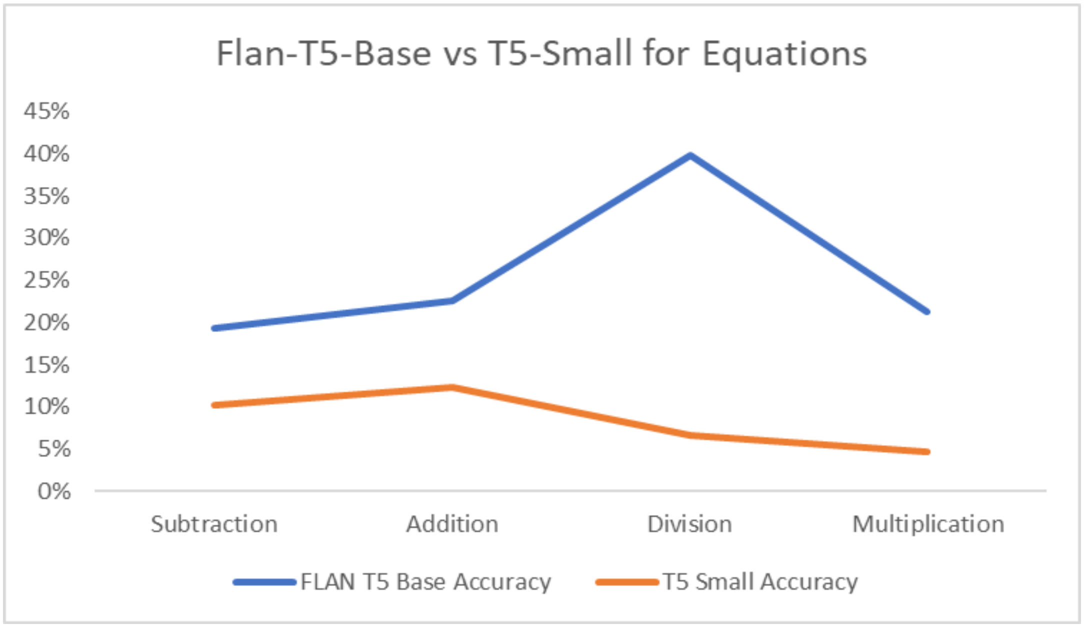
ROUGE scores of ROUGE-1 (0.605), ROUGE-2 (0.287), and ROUGE-L (0.605) evaluated the model's linguistic performance in reproducing unigrams, bigrams, and maintaining coherence.

Figure 3: Bar Chart of ROUGE Scores for T5 Base



Comparing Flan T5 Base and Flan T5 Small, Flan T5 Base (23.6% accuracy) significantly outperformed T5 Small (9.4%), indicating the pivotal role of model size and capacity in achieving higher accuracy rates across arithmetic operations. Flan T5 Base's larger size and complexity allowed it to capture intricate patterns better.

Figure 4: Performance Differences between T5 Base and T5 Small



Both models, however, faced challenges in reproducing consecutive word sequences accurately (ROUGE-2 scores). The equation-answering model displayed higher accuracy than the numerical-focused model, indicating potential complexities in precise numerical computation and prompting further examination of the model's numerical reasoning capabilities for future enhancements.

Conclusions

The project underscores the effectiveness of transformers and LLMs in translating word problems into equations. While leveraging both T5 models, the Flan T5 base exhibited stronger performance in ROUGE scores and achieved an accuracy of 23.6%. Conversely, the T5 small model maintained consistent accuracy across various operations, while the Flan T5 model excelled particularly in division operations. Despite reaching a satisfactory accuracy of 24%, there remains room for improvement. The limitations of the Flan T5 model with more complex problems indicate the need for improvements in model capabilities. Evaluating other transformer variants could offer insights for addressing this challenge.

Original Code Percentage

Approximately 41% of code used was sourced from the Internet.

References

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.

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.

ROGUE REFERENCE: https://medium.com/@priyankads/rouge-your-nlp-results-b2feba61053a

T5 Reference: https://exemplary.ai/blog/flan-t5