Modeling Symbolic Mathematics with Transformers

Sri Ram Bandi
sbandi@umass.edu

1 Introduction

Transformers from Vaswani et al. (2017) have shown great promise in the sequence transduction, language generation tasks, and much more, and generally emerged as the leading architecture of any Natural Language Processing task. In this project, I would like to explore its application to an interesting dataset of mathematical expressions, with hierarchical structure. Creating the dataset, transforming the dataset into trainable sequences and the inference technique used are interesting to work with in this project.

2 Related work

Few interesting previous works have tackled the problem of modeling hierarchically structured datasets. Of them, Lample and Charton (2019), Murty et al. (2023) and Nandi et al. (2024) were of particular interest.

Murty et al. (2023) and the works it depends on, goes onto explain the phenomenon of *structural grokking*, where the transformer language models can learn to generalize hierarchically after training for extremely long periods, far beyond the point when in domain accuracy has saturated. It also explains the relation of this phenomenon's efficacy to that of the depth of the model, basically exhibiting a inverting U-shaped relation, where the intermediate depth models generalise to structural data better than very deep and very shallow transformers.

Nandi et al. (2024) talks about softly injecting syntactic(structured) inductive biases into the transformer model. They propose an auxiliary loss function, called *TreeReg*, in addition on top of the standard language modeling objective. They show, that when a model is pre-trained with *TreeReg*, the model generalises better to OOD data for structured tree like data. This also worked, when a model pre-trained with a standard LM objective, is then finetuned with *TreeReg*, it improves its generalisation capabilities as well.

This project tries to emulate the work done in Lample and Charton (2019) where they directly solve this problem on this particular dataset. They laid out the methodologies to create a mathematical expressions dataset of various math domains. They train this symbolic mathematics dataset with the transformner model, and at evaluation time use Beam Search to evaluate the model's output. Their key idea in modeling a hierarchically structured data like this, is to transform them into linear sequences, train and use inference on them.

3 Your approach

Although Murty et al. (2023)'s *structural grokking* helps generate well on hierarchically structured outputs, this method needs to have both non-structured and structured data for achieving it. It also needs to be trained for long periods of time, well past what the other two works have suggested. Whereas, Nandi et al. (2024)'s work seems quite useful for our task, with only needing to add the *TreeReg* auxiliary loss function to regular training regime, flexibly at pre-training time or finetuning time.

In my approach in modeling this dataset, I will follow the work of Lample and Charton (2019) in creating the symbolic math expressions, transoforming them into linear sequences to be fed to the model for training and inference - which let's call the *Base* approach. I would like to then evaluate the outputs at inferecne using *Beam Search*. If time permits, I'd like to finetune the pre-trained model obtained with the *Base* approach with the *TreeReg* loss function. I'd like to then compare the accuracy of the *Beam Search* evaluation method on varying window sizes from 1 to 4, on both the

Base approach and the *Base* + *TreeReg* approach.

What baseline algorithms will you use? For the baseline algorithm, I will use *SymPy* python library to get the solution of a mathematical expression. I can then evaluate the output at inference by comparing the baseline and model outputs. The comparision here is not so straight-forward, because a mathematical expression obtained as an output at inference can be:

- *one of* the many valid results the prompt math expression can yield.
- further simplified to match the results of the *Baseline* algorithm.

So, we would need to use *Beam search* with varying window lengths to evaluate our model output against the baseline algorithm.

3.1 Schedule

This project has tasks encompassing the whole range of an NLP model lifetime, from dataset creation, model implementation, training to evaluation. I plan to schedule the tasks as below:

- **Dataset:** Develop dataset generation scripts, that create the mathematical expressions of binary and unary + binary(mixed) expressions; get their corresponding solutions using sympy; and transform the expressions to a trainable sequences dataset - 2 weeks; *Done* - https://github.com/srirambandi/compsci685).
- **Transformer Model:** Implement the transofrmer model for training and inference - 2 weeks.
- **Evaluation implementation:** Implement the *Beam Search* algorithm to be used for evaluation - 1 week.
- **Training and Inference:** Do training and inference runs on different compositions of the dataset - 2 weeks.
- **Analysis:** Collate the results from the experiments and analyse evaluation metrics, chart them out - 1 week.
- *If time permits, finetune the pre-trained model with TreeReg and compare the methods.*
- **Final Report:** Work on the final report - 1 week.

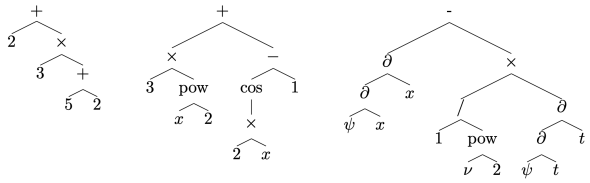

Figure 1: Mathematical expressions as trees, a one-to-one map between the expressions and the trees

4 Data

Data for this project is unlike other textual data which scraped from internet or proprietarily sourced. The nature of this data is different to natural language data generally used with Transformer language models as well. The dataset for this project is a programmatically generated mathematical expressions of various math domains(algebra, trigonometry, calculus), that are treated as expression trees, as shown in the Figure 1, and are then transformed to unique tree to sequence mappings to be used for training the model.

The transformation from expression trees to sequences are infix notations or in-order traversals of the expression trees. For example, the above three expression trees are one-to-one mapped with the following transformed sequences.

- + 2 x 3 + 5 2
- + x 3 pow x 2 - cos x 2 x 1
- - δ δ ψ x x x / 1 pow v 2 δ δ ψ t t

In this one-to-one mapping of expression trees to sequences, we can transform from one type to other. So, when forming a training dataset or inference prompt, we transform an expression tree to a sequence, as above examples. And, during inference, we transform the sequence to an expression tree for human readability.

The dataset can be acquired by writing the program for randomly generating the expressions, and then generating their corresponing solution pairs, using a set of algorithms mentioned in the original paper. By controlling the number of nodes in the expression tree and the number of different unary and binary operators and operands, we can essentially control the complexity of the mathematical expression and its solution and there so, we can chose to train various levels of difficulty datasets.

5 Tools

In this project, we will be using the following libraries and codebases for their following purposes:

SymPy

To generate and check solutions for some simple mathematics expressions in the dataset.

PyTorch and Huggingface

For building the transformer model, loss functions, optimisers, sequence embeddings, encoding and decoding. These libraries are used at preprocessing stage, to tokenize the data, train the models, and at inference stage as well.

Matplotlib, NumPy

For analying the results, charting loss values, evaluating the inference procedures for multiple Beam Search windows etc.,

Google Colab?[1] and other GPU services

For training the transformer model and during inference.

6 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - No

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
  - your response here
- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?
  - your response here

[1] As we said in class, we strongly suggest https://colab.research.google.com!

References

Lample, G. and Charton, F. (2019). Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*.

Murty, S., Sharma, P., Andreas, J., and Manning, C. D. (2023). Grokking of hierarchical structure in vanilla transformers. *arXiv preprint arXiv:2305.18741*.

Nandi, A., Manning, C. D., and Murty, S. (2024). Sneaking syntax into transformer language models with tree regularization. *arXiv preprint arXiv:2411.18885*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, , and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.