P-GLAM: GENERATING VALID ARITHMETIC EXPRESSIONS WITH GPT-2 *

Yashovardhan Srivastava
National Institute of Technology, Warangal
Warangal, Telangana
ys21chb0a57@student.nitw.ac.in

ABSTRACT

In this paper, we explore the capabilities of Language Models in generating valid arithmetic equations without any domain knowledge. Inspired by the famous Infinite Monkey Theorem, we create a dataset of arithmetic equations and train a GPT-2 inspired model, which we call P-GLAm, on this dataset. We then evaluate the equations generated by P-GLAm using the Sympy evaluator to determine their validity. Our findings demonstrate that Language Models have the potential to generate valid mathematical expressions without any domain knowledge, and our analysis sheds light on the theoretical limits of these models. In this field of Natural Language Processing and Probability Theory, this research provides insight into the potential applications of Language Models in fields of symbolic operations.

***K*eywords** Language Models · Symbolic Operations · GPT

Introduction

In recent years, the development of Language Models has revolutionized the field of Natural Language Processing. These models have shown remarkable performance in various tasks such as machine translation, text summarization, and language generation. However, the extent of their capabilities remains an open question, particularly when it comes to generating valid arithmetic equations, and other symbolic operations, their capabilities are limited, and are prone to several downfalls.

To explore the potential of Language Modeling in this area, we draw inspiration from the Infinite Monkey Theorem, a famous experiment in probability. The theorem suggests that a monkey hitting keys at random on a typewriter keyboard for an infinite amount of time will almost surely type complete works of Shakespeare. This is a probabilistic argument, and while the probability of this occurring is extremely low, it is technically not zero.

The Dataset

The arithmetic equation dataset used in this research was generated by randomly combining digits (0-9) and binary arithmetic operators. No external filters were used to remove inappropriate language, as the dataset was limited to this restricted domain of arithmetic equations. The assumption made was that the model would be able to learn the patterns and rules of arithmetic equations solely from this dataset.

To provide more context, we can draw parallels to the datasets used for language models, such as the Colossal Clean Crawled Corpus. However, unlike these datasets, we did not need to apply external filters for inappropriate language. Instead, our language was limited to a universal and non-offensive domain of arithmetic equations.

To conduct our experiment, we first create a dataset of arithmetic equations composed of binary arithmetic operators and digits from 0 to 9. We then train a GPT-2 inspired model, which we call P-GLAm, on this dataset. The architecture

* *Citation*: **Authors. Title. Pages.... DOI:000000/11111.**

of P-GLAm follows the transformer-based architecture of GPT-2, but with modifications tailored for the arithmetic domain.

We then use P-GLAm to generate random arithmetic expressions, which we evaluate using the Sympy evaluator. By evaluating the equations generated by P-GLAm using the Sympy evaluator, we can determine whether the equations are valid or not.

To measure the success of our experiment, we repeat it several times and calculate the probability of P-GLAm generating a valid arithmetic equation. We define a valid arithmetic equation as one that has a single numeric value and is mathematically correct. Our analysis includes measuring the accuracy of P-GLAm, analyzing the structure of the generated equations, and comparing the results to the probability of a monkey typing a valid equation according to the Infinite Monkey Theorem.

Model Architecture

P-GLaM is built upon the successful architecture of the GPT-2 model developed by OpenAI. The model is designed with a Transformer architecture, consisting of key components such as single headed attention, multi-headed attention, and layer normalizations, an overview of which is given in the figure below. To optimize the model's performance, layer normalizations were incorporated after each sub-block and after the final self-attention block.

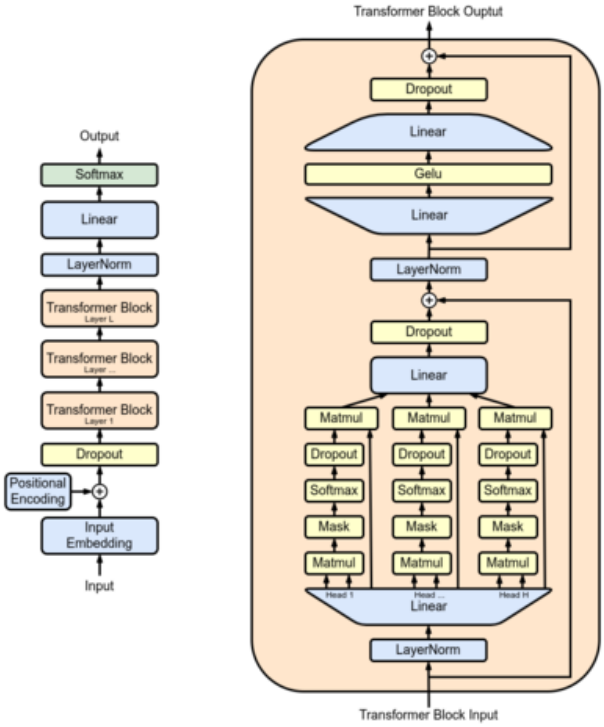

Figure 1: GPT Model Architecture

For the baseline model, the vocabulary size was limited to 15 tokens, including digits from 0-9 and five basic arithmetic operators. Additionally, the context length was restricted to 10 tokens. These design choices ensure that the model can handle the limited domain of arithmetic equations while maintaining a high level of accuracy.

Training

To evaluate the performance of our proposed P-GLAm model, we trained a 3.87M parameter baseline model on a Tesla T4 16 Gigabytes GPU. The training process was carried out using Google Colaboratory, and it took approximately 7 minutes to complete.

To train the model, we used the P-GLAm dataset that we generated, consisting of 88629 tokens. We used the parameters described in the table and trained the model for 500 steps for a total of 7 minutes. The Adam optimizer with a learning rate of 0.0003 was used, with standard values of β_1, β_2, and ϵ.

Hyperparamter	Value
Context Length	10
Batch Size	16
Evaluation Interval	100
Embedding Dimensions	128
Dropout	0.2
Layers	20
Head	20
Iterations	500
Evaluation Iterations	50
Learning Rate	0.0004

Table 1: Hyperparamter Reference Table

During the training period, we used Residual dropout with the probability of 0.2. The purpose of using Residual dropout is to prevent overfitting and ensure the generalization ability of the model.

Results

The probability of correctly solving mathematical problems using the P-GLaM model improved from 0.54 to 0.59 for a dataset of 5000 tokens. However, since only a small fraction of the data generated by P-GLaM was used for inference, it is difficult to draw a definitive conclusion. With improved computational capabilities, we will be able to conduct more rigorous testing and draw more confident conclusions about the model's performance on symbolic operations and manipulations.

Conclusion

Although our results are somewhat inconclusive due to limited hardware capabilities, our initial tests are promising. Despite only being able to infer on a small sample of 5000 tokens generated by P-GLaM, we observed an improvement from 0.54 to 0.59 in the model's ability to understand the semantics of algebraic expressions. These findings demonstrate the potential of P-GLaM as a language model capable of symbolic manipulation.

Acknowledgments

This project has been partially supported by myself, as a benefactor and sponsor. It is my hope that this research will contribute to a better understanding of the symbolic manipulation capabilities of language models.