TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Major Project Report

On

Linear Math Word Problem with T5 Model and Policy-based RL

Submitted By:

Nikhil Pradhan (41771)

Nistha Bajracharya (41773)

Prinsa Joshi (41780)

Samman Babu Amgain (41787)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March 2024

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Major Project Report

On

Linear Math Word Problem with T5 Model and Policy-based RL

Submitted By:

Nikhil Pradhan (41771)

Nistha Bajracharya (41773)

Prinsa Joshi (41780)

Samman Babu Amgain (41787)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus, Kathmandu, Nepal

In the partial fulfillment for the award of the

Bachelor's Degree in Computer Engineering.

Under the Supervision of:

Er. Bibat Thokar

March 2024

DECLARATION

We hereby declare that the report of the project entitled **"Linear Math Word Problem with T5 Model and Policy-based RL"** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Computer Engineering**, is a bona fide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Nikhil Pradhan (Class Roll No: 076/BCT/022) _____

Nistha Bajracharya (Class Roll No: 076/BCT/024) _____

Prinsa Joshi (Class Roll No: 076/BCT/031) _____

Samman Babu Amgain (Class Roll No: 076/BCT/040) _____

March, 2024

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to **the Department of Electronics and Computer Engineering, IOE, Thapathali Campus,** a major project work entitled **"Linear Math Word Problem with T5 Model and Policy-based RL"** submitted by **Nikhil Pradhan, Nistha Bajracharya, Prinsa Joshi** and **Samman Babu Amgain** in partial fulfillment for the award of Bachelor's Degree in **Computer Engineering**. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Computer Engineering.

_____

Project Supervisor
Er. Bibat Thokar
Department of Electronics and Computer Engineering, Thapathali Campus

_____

External Examiner
Er. Himalaya Kakshapati

_____

_____

Project Co-ordinator
Mr. Umesh Kanta Ghimire
Department of Electronics and Computer Engineering, Thapathali Campus

_____

Mr. Kiran Chandra Dahal
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus
March, 2024

COPYRIGHT

ACKNOWLEGEMENT

We would like to express our sincere gratitude towards the Institute of Engineering, Tribhuvan University for the inclusion of the major project in the course of Bachelor in Computer Engineering. We are also grateful to the Department of Electronics and Computer Engineering, Thapathali Campus for providing us with the resources and support which is needed for this project. We would like to convey our gratitude to Er. Bisikha Subedi and Er. Bibat Thokar, our project supervisors, for their constant support. We also like to thank the creators of the research articles that we used as references for this project. Finally, we would like to thank all our teachers and our friends for motivating us and assuring their full support in challenging times ahead.

Nikhil Pradhan (Class Roll No.: 076/BCT/022)

Nistha Bajracharya (Class Roll No.: 076/BCT/024)

Prinsa Joshi (Class Roll No.: 076/BCT/031)

Samman Babu Amgain (Class Roll No.: 076/BCT/040)

ABSTRACT

This project aims to create a system that solves algebraic linear math word problems involving one and two variables into accurate mathematical equations with their solution. A diverse dataset was gathered from sources; Kaggle and GitHub, covering various scenarios like age, arithmetic word problems, geometry and number problems. Through the use of Proximal Policy Optimization (PPO) Reinforcement Learning algorithm, and transformer models T5, the intricate relationships between natural language text and mathematical concepts are understood by the system. Kaggle was utilized for employing the T5 transformer model to translate natural language descriptions into mathematical equations. Within an RL environment, the model undergoes optimization using PPO, thereby enhancing its equation-generation accuracy over time. Leveraging the capabilities of the mathsteps Node package, the system extends its functionality to solve algebraic linear math word equations involving up to two variables. The model's performance was evaluated using metrics such as Equation Matching (EM), ROUGE-1, ROUGE-2, ROUGE-L, and answer comparison. It achieved an exact match accuracy of 73.36% and final answer comparison accuracy of 76.38%. For ROUGE-L, precision, recall, and F-measure scores of 0.9371, 0.9266, and 0.9294 were achieved respectively. These accurate results demonstrate the model's proficiency in accurately formulating equations from diverse math word problems.

Keywords: Algebra, Linear Equations, Machine learning, PPO, Problem-solving, RL environment, T5 model

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformer
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Units
LHS	Left-Hand Side
LSTM	Long-Short Term Memory
MDPs	Markov Decision Process
MLE	Maximum Likelihood Estimation
MSE	Mean Squared Error
NLP	Natural Language Processing
PPO	Proximal Policy Optimization
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RHS	Right-Hand Side
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
T5	Text-To-Text Transfer Transformer
Uni LM	Unified pre-trained Language Model
WMT	Word Memory Test

1. INTRODUCTION

1.1 Background

Algebraic math word problems are essential in math education because they require individuals to use algebraic concepts to translate verbal descriptions into mathematical equations. Solving such problems improves math skills, critical thinking, and problem-solving abilities, and they cover a wide range of scenarios such as age, distance, and mixing challenges. Many people, however, struggle because they have difficulty understanding terminologies, identifying mathematical structures, or constructing proper equations. To address these challenges, many tools and strategies have been developed, including classic problem-solving procedures and instructional methodologies. Educators teach problem-solving strategies and provide opportunities for practice.

In recent years, technology and artificial intelligence have played an important role in the solution of algebraic arithmetic word problems. Researchers are investigating machine learning and natural language processing to develop automated systems that can understand and solve problems. Reinforcement learning advancements improve these solvers over time, and AI-driven solutions can address frequent challenges encountered by learners. Automated linear math word problem solutions provide accessible and timely assistance, minimizing dependency on human support. They are effective teaching tools because they allow students to practice problem-solving skills on their own. Furthermore, AI-driven solutions contribute to the improvement of instructional technology by encouraging research in education and adaptive learning systems, resulting in a more efficient and engaging learning experience.

1.2 Motivation

The ambition to improve the usability and effectiveness of education in mathematics serves as the driving force behind the creation of a linear math word problem solver. It is well recognized that linear word problems present a considerable challenge for many students, frequently resulting in dissatisfaction and a lack of faith in their mathematical skill. By developing an automatic solver, students can be given a useful resource that

will aid in their educational endeavors. Such a solver enables students to improve their problem-solving abilities and expand their grasp of a linear concepts.

Additionally, a linear math word problem solution may be able to consider each student's unique requirements and learning preferences. It may adjust to various skill levels, offer customized scaffolding or clues, and provide opportunities for practice and reinforcement. In the end, the creation of an efficient and approachable linear math word problem solution can help to enhance mathematics education outcomes, boost student confidence, and foster a lifetime appreciation for the subject.

1.3 Problem Definition

The issue to be solved is the difficulty students and individuals have in completing linear math word problems. Linear word problems require the translation of written descriptions into mathematical equations and expressions, providing challenges in language comprehension, mathematical reasoning, and problem-solving abilities. Many students struggle to understand the problem descriptions, extract essential information, and create adequate mathematical representations. This leads to frustration, a lack of confidence, and stifled development in mathematical instruction. The goal of our project is to create an automated algorithm that can decipher and resolve a variety of Linear word problems. The solver must be able to examine the written descriptions, pick out important details, and produce precise mathematical equations or expressions. By developing such a solver, accessibility, effectiveness, and enjoyment are increased for students and individuals, enabling them to hone their problem-solving abilities and lay a solid basis in linear reasoning.

1.4 Objectives

The main objectives of the project are:

- To generate mathematical linear equations from math word problems involving one and two variables

- To provide user-friendly platform to input word problems and effectively receive solutions for those problems.

1.5 Scope and Application

Math word problems are huge challenges for students and frequently hinder their learning progress. The need to comprehend and formulate equations based on textual explanations adds to the complexity, causing challenges with application, critical thinking, and problem-solving. Traditional approaches to teaching and solving math word problems are not always helpful, and many people fail to build precise equations, which has an impact on their overall mathematical aptitude. To address this problem, creating a Math Word Problems Equation Generator using advanced natural language processing and reinforcement learning techniques. To interpret and generate textual representations of arithmetic word problems, T5 is used. T5 has demonstrated outstanding performance in a variety of language-related activities and is well-suited for our application.

The project can only tackle problems that can be written as linear equations. Equations involving variables raised to the first power only will be solved. The project is intended to handle math word problems with linear equations with up to two variables. The main focus is on age-related problems, athematic word problems, speed-distance-time, basic geometry and number problems. Model will generate equations involved up to two variables, denoted as x and y specifically. The solver outputs the solutions to the equations generated from the word problems. Users will receive the values of the variables (x and y) that satisfy the given conditions. The user interface is intuitive and user-friendly, allowing users to input the word problems easily. The output is presented clearly, showing the steps taken to solve the equation and the final answer.

The application of this project is multi-fold and can benefit various stakeholders:

1.5.1 Researchers and Developers

The project contributes to the advancement of artificial intelligence and natural language processing in the context of algebraic math word problem solving. It investigates the capabilities and constraints of machine learning models for answering algebraic word problems effectively without the need for lengthy explanations, which might provide insights and recommendations for future study and development in the subject.

1.5.2 Professionals and Individuals in Real-Life Applications

Professionals and individuals in disciplines requiring algebraic problem solving, such as engineering, finance, or data analysis, can use the solver. It can provide a quick and easy way to solve algebraic word problems, allowing experts to focus on higher-level analysis or decision-making activities.

1.5.3 Educational Technology Developers

The creation of an algebraic math word problem solver that delivers solutions without providing step-by-step explanations can serve as a foundation for the creation of more complex educational technology applications. This includes incorporating the solver into intelligent tutoring systems, adaptive learning platforms, or gamified learning applications, which improves the entire learning experience and promotes autonomous problem-solving skills.

1.5.4 Students

The algebraic math word problem solution can be a useful learning tool for students of all academic levels. It can help them grasp and solve algebraic word problems. The solver can assist students in developing problem-solving skills, improving mathematics comprehension, and developing confidence in dealing with algebraic word problems.

1.6 Report Organization

The material present in the report is organized into seven chapters. The first chapter is an introduction section, which provides information related to linear math word problems, it's background, scope and application. Objective and problem definition of the project are detailed in this section. The second chapter, literature review, previews a summary of researched works that have been previously carried out in the field related to our project along with their drawbacks and limitations. The third chapter defines the requirement analysis which consists of project requirements and feasibility analysis. The fourth chapter defines the project system, it's architecture, relevant block diagrams, algorithms, and methodology implied in the project. Activity diagram, state diagram, class diagram, use case diagram, ER diagram and data flow diagram are described in this section. In fifth chapter, implementation architecture and working principle of the

project are elaborated. The sixth chapter contains the result of the project. Here, the result is displayed and various conclusions are derived and discussed from the result obtained. The seventh chapter contains the possible future enhancement of this project. The eighth chapter concludes the project report.

2. LITERATURE REVIEW

Research on solving word-based math problems by computers has been going on since the start of 1963. Recent advancement in machine learning has created an open road of development. Current State of the art technique use seq2seq model for understanding the problem and generating the appropriate equations.

Machine Guided Solution to Mathematical Word problems, the study focused on solving elementary math word problems by applying classification algorithm to classify the sentence and assign it as one of Join-Separate, Part-Part Whole or Compare type [1]. Cascading of the necessary classifiers was done based on the problem type of the given word problem. Specifically, in Join-separate problem, firstly the sentence was classified into a functional type, then a sign prediction was done. For the classification and sign prediction, Random Forest algorithm was applied, and the equation generation used rule-based deductive learner that combined result of classification to establish numerical values required in the equation.

In 2017, the Transformer, the first sequence transduction model that purely rely on attention mechanisms was proposed [2]. It effectively replaced the conventional recurrent layers. This novel architecture drew global relationships between the input and output sequences using encoder-decoder structures with multi-headed self-attention. The Transformer showed that it could be trained noticeably faster than traditional designs based on recurrent or convolutional layers because it only used attention techniques. Additionally, it performed at a cutting-edge level on the WMT 2014 English-to-German and English-to-French translation tasks. The Transformer's success signaled a significant development in sequence-to-sequence modeling and demonstrated the effectiveness of attention-based methods for tasks involving natural language processing.

Deep Neural Solver for Math Word Problems, direct translation of the math word problem into equation templates using an RNN instead of using previously used statistical learning approaches was proposed [3]. They developed an RNN-based seq2seq model to generate equations from math problems. A significant number identification model was also implemented to check whether a number in a problem

were to appear in the corresponding mathematical equation. Gated Recurrent Units (GRU) was used as encoder for the model. The redesigned activation function was used instead of SoftMax to prevent mathematical inconsistencies in the generated equation template (e.g., x = n1 + + n2). So, 5 rules were used to prevent inconsistencies from occurring. The model consisted of one word embedding layer, two-layer GRU as encoder and two-layer LSTM as decoder. LSTM-based binary classifier was used to find whether a number in the given word problem was significant or not. The retrieval model was established as a comparison for the developed model. The retrieval model solved the problem by using lexical similarity between the testing problem and each problem in the training data and applying the most similar equation template to the test problem. Use of some other methods on small data sets could be used to increase accuracy further.

Neural Math Word Problem Solver, A reinforcement learning approach to increase the accuracy of the result was proposed [4]. All the previous models had used probabilistic Maximum Likelihood Estimation (MLE). Spurious number generation and number generation in the wrong place were the major problems on previous model. To overcome this problem, copy and alignment mechanism was implemented on the seq2seq model to avoid these problems. Copy made so that the model only copied the numbers from the given problem thus avoiding the problem of generating spurious numbers. Alignment helped to align information between the problem and the generated equation. For increasing the accuracy of the solution, policy gradient algorithm was used instead of MLE as MLE suffered from "train-test discrepancy". The model was combined with traditional feature-based model by passing the result of the neural model as a feature to the feature-based model. Thus, created model got the best of both worlds increasing the accuracy of the model. The model outperformed all the other models following the hybrid approach and applying Reinforcement Learning.

Template-Based Math Word Problem Solvers in 2019, was designed to solve arithmetic math word problems [5]. The model used a seq2seq model to predict a tree-structure template. The resulting tree's leaf node was inferred numbers and unknown operators were kept in the inner nodes of the tree. The unknown operators of the tree were then inferred by using a Bi-LSTM in bottom-up approach. The tree was used for the template creation of the respective math problem. The model could not perform well on problems

that would generate long templates. Using modern NLP, Semantic understanding could also be improved.

In 2019, a math word problem solver that uses a transformer with two decoders working in opposite directions was suggested [6]. The model outperformed the RNN approach of Copy and Align. In both the inquiry and the equation, the number tokens were changed to special symbols. The transformer was then trained to produce equations using those symbols without being shown actual values. The two decoders enhanced encoder training. Reinforcement learning was also utilized to increase the number of accurate answers by solving produced equations rather than generating exact duplicates of equations in training data. It concludes that a Transformer-based system may create equations for arithmetic word problems. Reinforcement learning can also improve performance.

A novel approach to solving math word problems was introduced by Qinzhou Wu that focused on explicit numerical values [7]. This was a new approach to previously used template-based methods that substituted tokens in the equation rather than using the given number to generate a unique equation. Experiments were conducted on two large-scale datasets Math23K and Ape210k. The model was called NumS2T which enhanced performance by explicitly incorporating numerical values into sequence-to-tree network. Attention based sequence to tree model with problem encoder and tree structured decoder was used to create math equations. Numerical values were explicitly incorporated so that number-aware problem representations could be achieved. The use of self-attention mechanism improved the accuracy by 8.8% and 0.7% in Math23k and APE210k as it helped to model the relations between the numerals.

A Deductive Reasoning Model for Math Word Problem Solving" presents DEDUCTREASONER [8], a novel deductive reasoning model designed to enhance math word problem solving. The paper demonstrates the model's superiority over existing approaches, achieving significant improvements in accuracy across multiple benchmark datasets. The incorporation of rationalization further enhances the model's performance and transparency. The findings contribute to the field of automated math problem solving by showcasing a state-of-the-art approach that utilizes pre-trained language models.

In the study "Attending to Mathematical Language with Transformers," researchers employed the transformer architecture to generate, evaluate, and use mathematical expressions to train neural network models [9]. The expressions and their related targets were considered as character-level sequence transduction tasks, with both the encoder and decoder applying attention mechanisms. The results show the transformer's ability to understand and evaluate symbolic variables and expressions provided as character strings. Notably, the transformer excelled in addition, subtraction, and multiplication with positive and negative decimal numbers.

In 2020, a unified framework that transformed different text-based linguistic issues into a standard text-to-text format was introduced [10]. The model used the Colossal Clean Crawled Corpus for pre-training, and it produced outstanding results on many benchmarks covering summarization, question answering, text categorization, and other tasks. It treated every processing problem as a text-to-text problem. A text-to- text encoder-decoder transformer model was specifically designed for transfer learning called T5.

In 2022, a strategy which solved arithmetic word problem issues using a combination of unsupervised pretraining and supervised tuning strategies was proposed [11]. The work used language modeling targets on unlabeled data to build starting parameters, which were subsequently adapted to the specific task using supervised learning. UniLM performed well in unidirectional, sequence-to-sequence, and bidirectional prediction tasks. According to the findings of the study, as the complexity of the sentences increased, the performance of all models decreased.

In 2023, using GPT-3 to classify word problems, extract equations, and generate word problems was proposed [12]. Math word problems dataset focused on systems of two linear equations which were variable in size; 20, 30, 50, 100, 200, and 1000 problems. GPT-3, from OpenAI with 175 billion parameters, was used. It claimed that transformers can capture longer-term dependencies within text to grasp context and are trained faster. It has been discovered that the number of instances given in the prompt influences the model's performance on the same task. It concludes that fine-tuned models outperform few-shot learning when it comes to extracting systems of linear equations.

The minor errors missed by sequence-to-sequence models is was addressed by introducing a ranking task which complemented generation with error detection [13]. It proposed the solution of generator and ranker. Generator was based on pre-trained models; BART which generates multiple candidate expressions and the Ranker was used to distinguish correct expressions from incorrect ones. Since it used Chinese dataset; Math23k it used mBART25 which is a multilingual BART. It showed 7% improvement on Math23k and achieving State-of-the-art performance. It also provided robustness to unseen problems and complex expressions.

In conclusion, the findings of the study imply that combining T5 (Text-to-Text Transfer Transformer) and reinforcement learning has the potential to improve the accuracy and efficacy of algebraic math word solvers. T5, a cutting-edge language model, has exhibited outstanding performance in a variety of natural language processing tasks, including text production and comprehension. T5 can potentially improve the model's capacity to interpret issue descriptions and create relevant equations by applying it to algebraic math word problems.

Furthermore, the use of reinforcement learning is an effective method for refining and optimizing the solver's performance. The model can learn from trial and error by utilizing reinforcement learning techniques, boosting its capacity to provide correct and coherent solutions. This method addresses issues such as inaccurate number production and the right arrangement of numbers within equations. T5 combined with reinforcement learning has the advantage of using the power of deep learning models while combining feedback and assistance to improve problem-solving capabilities. It has the potential to improve accuracy while overcoming the constraints of prior techniques.

3. REQUIREMENT ANALYSIS

3.1 Project Requirement

3.1.1 Hardware Requirements

- **GPU**: A powerful GPU, such as NVIDIA GeForce RTX or Tesla series, is recommended to accelerate training and inference processes, as transformer models like T5 can be computationally intensive for fine tuning the model.

3.2.1 Software Requirements

- **Python**: Python is a popular programming language widely used in the machine learning and NLP community. Python is used for implementing the algorithms, data preprocessing, model training, and evaluation.

- **Deep Learning Frameworks**: A deep learning framework that supports transformer-based models like PyTorch, and Hugging Face Transformers, were used as the major framework for model finetuning and model storage.

- **NLP Libraries**: NLP libraries such as transformers library from Hugging Face were used for text processing tasks like tokenization, parsing, and language modeling.

- **Reinforcement Learning Libraries**: RL has been implemented using the Transformer reinforcement learning toolkit available in the hugging face repository.

- **Data Processing and Visualization Libraries**: Pandas, NumPy, and Matplotlib were essential for data preprocessing, analysis, and visualization tasks.

3.2 Feasibility Analysis

3.2.1 Operational Feasibility

Because of its user-friendly interface, cross-platform compatibility, and capacity to handle many sorts of Linear Equation word problems, the project is operationally feasible. The solver is designed to analyze and solve issues efficiently without losing performance, making it scalable to handle rising usage. These qualities ensure that the solver may be easily incorporated into current processes and systems, resulting in a realistic and efficient solution for algebraic math word problem solving.

3.2.2 Economic Feasibility

The cost to develop this software is relatively low compared to other types of software due to access to open-source tools and platforms. No additional hardware is used, so the project is economically feasible.

3.2.3 Technical Feasibility

The technical feasibility analysis focuses on the availability of appropriate technologies and resources needed for the solver's development and implementation. The required transformers were available on Hugging face library. The initiative also guaranteed that appropriate computing infrastructure and resources were available to facilitate effective operation.

3.2.4 Legal Feasibility

The project's goals are to secure user data, avoid copyright infringement, and encourage fairness and openness in problem-solving. The project shows its legal feasibility and ensures compliance with applicable legislation and ethical standards by resolving these legal and ethical issues.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

Reinforcement Learning (RL) in conjunction with the transformer model T5 has been used to tackle the problem of generating equations from natural language text. The RL environment has been created where T5 agent learns to generate equations based on textual problem descriptions. The project encompasses training the T5 agent on equation-generation tasks and subsequently evaluating its ability to automate the translation of textual problems into mathematical expressions.

4.1 Transformer Architecture

The Transformer model architecture is a powerful neural network architecture that revolutionized various natural language processing tasks, such as machine translation, text generation, and question answering. It employs a self-attention mechanism to capture relationships between words in a sequence, allowing for parallel processing and effectively modeling long-range dependencies. The Transformer architecture can be used to analyze and understand problem statements written in natural language. For a math word problem expressed as a sequence of words, the Transformer model can analyze the problem statement, find related mathematical operations, and produce a solution. By encoding the input sequence with self-attention mechanisms, the model can identify word relationships, and differentiate between important and irrelevant information.

Here is a detailed explanation of the components and flow of the Transformer model architecture:

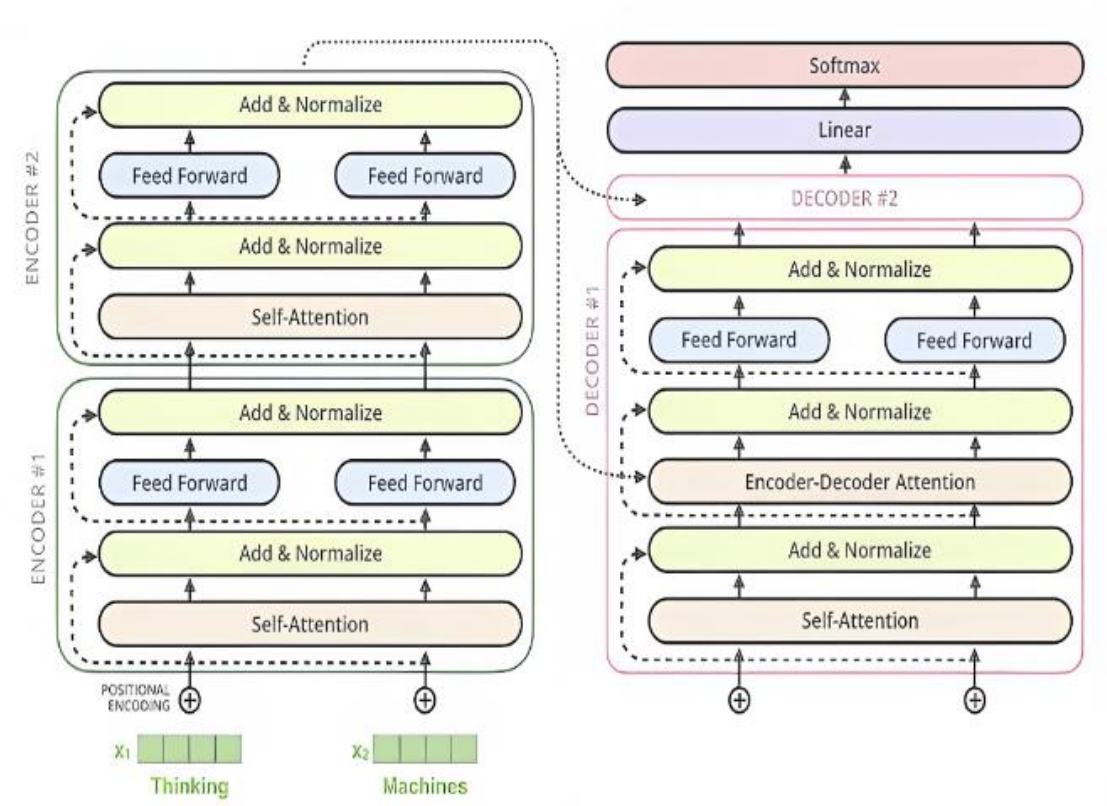


Figure 3- 1: Transformer Architecture Model [2]

4.1.1 Input Layer

The input to the Transformer model is a sequence of tokens representing the problem statement, such as "The sum of two numbers is 16, and their difference is 4. Find the two numbers". Every word in the input sequence is firstly positionally encoded to preserve the information of a word's position in the given sentence. To achieve positional encoding, sine, and cosine functions are used. Positional encoding using sine and cosine function is given by

$$P(k, 2i) = \sin\left(\frac{K}{n^{\left(\frac{2i}{d}\right)}}\right) \quad 4.1$$

$$P(k, 2i+1) = \cos\left(\frac{K}{n^{\left(\frac{2i}{d}\right)}}\right) \quad 4.2$$

Where:

k: position of the word in the input sequence

d: dimension of the embedded word

i: column index of the encoded value

n: user defined scalar

Each word/token is initially embedded into a continuous vector space through an embedding layer. The Positionally encoded vector is then added to the embedded vector to get the input to the encoder layer of the transformer.

4.1.2 Encoder Stack

The Transformer model consists of a stack of encoders, which are responsible for processing the input sequence. Each encoder contains two sub-layers a self-attention mechanism and a feed-forward network. Encoder is responsible for encoding the input sentence to produce a sequence of contextually enriched representations for each token. The input of the first encoder in the encoder stack is the embedded and positionally encoded input sentence. Encoder produces the output of this input by taking it through a series of steps. This output is then passed to the other encoder of the stack. The process goes on sequentially till the last encoder has performed its operations on its input. Output of the encoder stack is then passed to the decoder part of the transformer.

4.1.3 Self-Attention Mechanism

Self-attention allows the model to weigh the importance of different words in the input sequence when generating representations. It calculates attention weights for each word in the sequence by considering its relationship with all other words. This mechanism enables the model to capture contextual information and dependencies across the entire sequence simultaneously. In the self-attention mechanism of the Transformer model, three main components are involved Query (Q), Key (K), and Value (V). These components are used to compute the attention weights that determine the importance of different words in the input sequence.

Here's a detailed explanation of the Q, K, and V components in the self-attention mechanism:

- **Query (Q):**

The Query represents the word for which the attention weights is to be calculated. In the context of the Transformer model, the Query is derived from the input sequence and serves as the word of interest.

- **Key (K):**
  The Key represents the words that are compared to the Query to determine their relevance. In the self-attention mechanism, the Key is also derived from the input sequence and represents all the words in the sequence.
- **Value (V):**
  The Value represents the values associated with the words in the input sequence. In the context of the Transformer model, the Value is also derived from the input sequence and represents the information associated with each word.

The self-attention mechanism in the Transformer model operates by calculating the attention weights between the Query and each Key in the sequence. These weights determine how much importance should be given to each word (represented by its Value) when computing the output representation.

The attention weights are computed by taking the dot product between the Query and each Key and then applying a SoftMax function to obtain a normalized distribution. The resulting attention weights are used to weigh the corresponding Values, which are then combined to compute the output representation.

Example calculation of attention with small matrices for Q, K, and V:

$$Q = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad K = \begin{bmatrix} 7 & 9 & 11 \\ 8 & 10 & 12 \end{bmatrix}, \quad V = \begin{bmatrix} 0.1 & 0.3 & 0.5 \\ 0.2 & 0.4 & 0.6 \end{bmatrix}$$

Calculating the attention value using the formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad 4.3$$

where d_k is the dimension of the key vectors, where $d_k = 2$

Calculate QK^T:

$$QK^T = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} * \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Divide by $\sqrt{d_k}$:

$$\left(\frac{QK^T}{\sqrt{d_k}}\right) = \frac{1}{\sqrt{2}} \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Apply SoftMax:

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) = \begin{bmatrix} 0.04 & 0.09 \\ 0.36 & 0.91 \end{bmatrix}$$

Multiply by V:

$$A = \begin{bmatrix} 0.04 & 0.09 \\ 0.36 & 0.91 \end{bmatrix} * \begin{bmatrix} 0.1 & 0.3 & 0.5 \\ 0.2 & 0.4 & 0.6 \end{bmatrix} = \begin{bmatrix} 0.022 & 0.048 & 0.074 \\ 0.218 & 0.472 & 0.726 \end{bmatrix}$$

The final attention matrix A is:

$$A = \begin{bmatrix} 0.022 & 0.048 & 0.074 \\ 0.218 & 0.472 & 0.726 \end{bmatrix}$$

This represents the weighted combination of values in the matrix V based on the attention mechanism.

By utilizing the Q, K, and V components in the self-attention mechanism, the Transformer model is able to capture the relationships between words in the input sequence and effectively weigh their importance when generating representations. This mechanism enables the model to attend to the most relevant words for each Query, improving its ability to capture contextual information and dependencies within the sequence.

4.1.3 Multi-Head Attention

The self-attention mechanism is applied multiple times in parallel, each with its own learned weights called attention heads. The attention heads capture different types of relationships and provide multiple perspectives on the input sequence. This multi-head attention enhances the model's ability to capture different types of dependencies and improve performance.

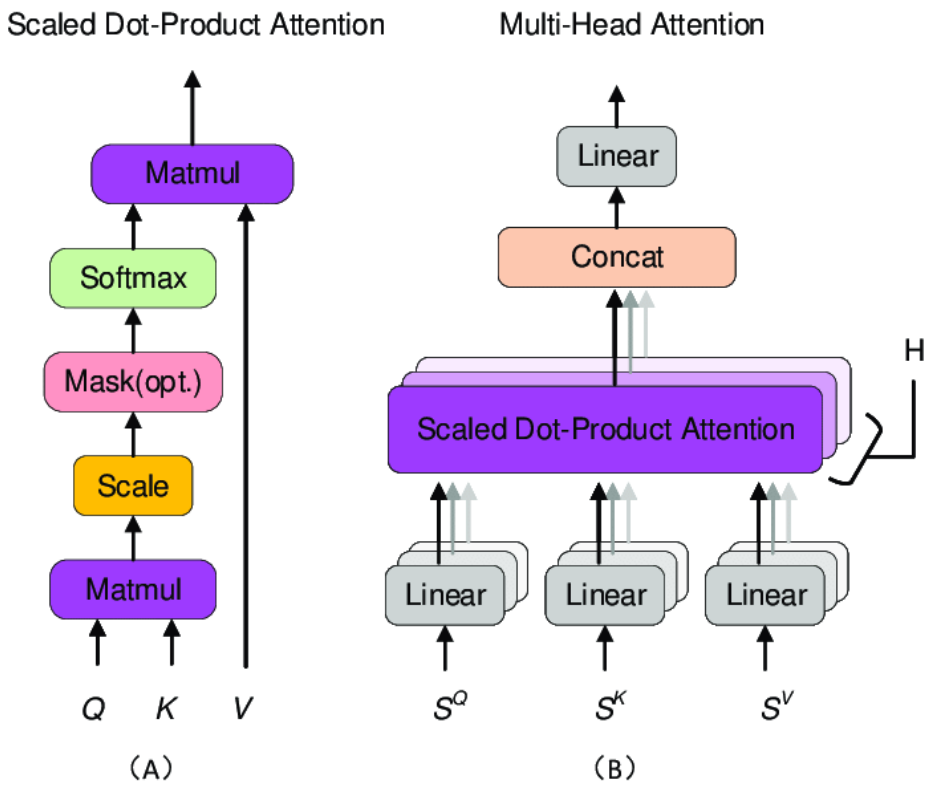


Figure 3- 2: (A) Scaled Dot-Product Attention and (B) Multi-Head Attention [2]

The total number of multi head attention on T5 base model is 144, which is given by given method:

Calculation:

Number of Layers: 12 (each containing both encoder and decoder components)

Number of Multi-Heads per Layer: 12

Total Number of Multi-Heads:12 layers × 12 heads/layer = 144 multi-heads

4.1.4 Feed-Forward Network

The output of the self-attention mechanism is routed through a feed-forward neural network to introduce non-linearity and mimic more complicated interactions between words. The feed-forward network is made up of two linear transformations separated by an activation function, often a Rectified Linear Unit (ReLU). The ReLU activation function is used to incorporate nonlinearity into the transformation, allowing the model to capture more intricate patterns and correlations in the data.

$$\mathrm{ReLU}(x) = \begin{cases} 0, \text{if } x < 0 \\ x, \text{if } x \geq 0 \end{cases} \qquad 4.4$$

The output of the self-attention mechanism is processed via the first linear transformation. This linear layer transforms the contextual representations of the tokens into a new vector space by applying a learned weight matrix to them. After the first linear transformation, a Rectified Linear Unit (ReLU) activation function is applied element-by-element to the resulting vector. By mapping all negative values to zero and keeping positive values intact, the ReLU activation function introduces non-linearity. The output of the ReLU activation function is then passed through another linear layer, where another learnt weight matrix is used to further alter the representations.

These linear transformations and the non-linear activation function are successfully combined by the feed-forward network to model the intricate relationships between the words in the input sequence. The relationship between the mathematical symbols, variables, and operations in the math word problem is better understood by the equation generator as a result of this procedure. The T5-based equation generator can efficiently learn and represent the complex patterns and dependencies contained in the math word problems by including the feed-forward network within each encoder.

4.1.5 Layer Normalization

Layer Normalization in the context of Transformers, such as the T5 math word problem solver, is a crucial technique used after every sub-layer, including feed-forward networks and self-attention mechanisms. This normalization process serves to stabilize

the model's training and enhance its performance by promoting quicker convergence and effective learning. the T5 model, layer normalization is applied to the output of each layer before passing the results to the next layer. This technique standardizes the outputs by dividing through the elements' standard deviation and subtracting the mean. By doing so, it mitigates the issue of internal covariate changes and allows the model to adapt more effectively to different math word problems.

The primary benefit of layer normalization in Transformers is its ability to improve the model's capacity for constructing accurate mathematical equations for a wide range of word problems. By stabilizing the distribution of the results, the model becomes more robust and less prone to overfitting or divergence during training. As a result, the T5 solver can generalize better to unseen data and produce more accurate solutions for various math word problems

Layer Normalization is a critical component in the design of Transformer-based models like T5, as it helps to enhance the solver's overall performance, stability, and adaptation to different math word problems. By normalizing the outputs of each layer, the model can learn more effectively and generate more precise mathematical equations for a wide range of problems.

4.1.6 Decoder Stack

The decoder stack is a key element of the math word problem equation generator utilizing T5, which makes it easier to generate algebraic answers based on the contextual information from the input sequence. The encoder stack and decoder stack are similar, but the decoder stack also has a multi-head attention layer operating over the encoder's output. The output sequence from the decoder stack is what represents the algebraic equation or answer to the provided math word problem. Like the encoder stack, the decoder stack uses the transformer design and the idea of self-attention to accomplish this.

The additional sub-layer of multi-head attention that is used in the decoder stack, however, is where the main distinction lies. This sub-layer enables the decoder to generate the solution while attending to the pertinent portions of the encoder's output. The decoder can successfully absorb and contextualize information from the input

sequence by paying attention to the encoder's output, ensuring that the generated equations are coherent and pertinent to the particular math word problem at hand.

The math word problem equation generator may produce precise and contextually appropriate algebraic solutions because the decoder stack incorporates multi-head attention over the encoder's output. This attention technique enables the decoder to concentrate on the data from the input sequence that is most important for solving the given math word problem, leading to more relevant and comprehensible equations. The model can efficiently comprehend and process the input text while generating accurate and meaningful solutions to a variety of arithmetic word problems thanks to the combination of the encoder and decoder stacks within the transformer architecture.

4.1.7 Output Layer

The Transformer model's final layer, the output layer, is vital in producing the textual answer to the algebraic math word problem. The representations of the model are then transferred through a linear transformation in the output layer after going through the decoder stack and getting the contextual representations for each token in the output sequence. To map the representations to the vocabulary space, which includes of all potential output tokens, including numbers, mathematical symbols, and words, this linear transformation adds learnt weights.

The output layer then uses the SoftMax activation function after the linear transformation. The SoftMax function normalizes the output token scores and transforms them into a vocabulary-wide probability distribution. This probability distribution shows the likelihood that each token will be the subsequent word or symbol in the math word problem's algebraic solution.

$$z_i = \sum_j W_{i,j} x_j + b_i \tag{4.5}$$

$$y = \text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_i)} \tag{4.6}$$

The math word problem equation generator efficiently produces the textual solution to the specified algebraic math word problem by applying the linear transformation and SoftMax activation in the output layer. The model can generate cogent and contextually appropriate algebraic answers for a variety of arithmetic word problems because it can predict the next token depending on the context of the preceding tokens.

4.2 T5 Model Architecture

The T5 architecture is based on the Transformer architecture. The Transformer architecture is a neural network architecture that uses self-attention to learn long-range dependencies between words.

The T5 architecture consists of two main components: an encoder and a decoder. The encoder is responsible for processing the input text and generating a sequence of hidden states. The decoder is responsible for generating the output text, one word at a time, by attending to the hidden states generated by the encoder. The encoder and decoder are both made up of a stack of self-attention layers. Each self-attention layer attends to all of the words in the input or output sequence and uses this information to back generate a new hidden state. Self-attention is a mechanism that allows a neural network to attend to different parts of an input sequence. In the case of T5, the input sequence is a piece of text. The self-attention mechanism allows the model to learn how the different words in the text are related to each other. The T5 base model has 12 encoder layers and 12 decoder layers.

By pre-training T5 on large-scale datasets containing a wide range of text transformation tasks, it can effectively capture language patterns and semantic information. The pre-trained T5 model can then be fine-tuned on specific downstream tasks to achieve high performance on various natural language processing tasks, including tasks related to word problem solving and equation generation.

4.3 T5 Model Fine Tuning

T5 Model being pre-trained on the C4 dataset can perform multiple tasks out of the box. T5 model can perform a wide variety of tasks such as language translation, text summarization, question answering and much more. Since pre-trained T5 can perform

well on Natural Language Understanding and Natural Language Generation tasks, it is well suited for the application for our math word problem solver.

For the T5 model to perform the task as intended, fine-tuning needs to be performed with a significant amount of data for the model to infer the solutions. Fine-tuning is the process of adjusting a pre-trained model to a specific task. It can be understood as training a general model to perform specific tasks. The proposed method for fine-tuning involves the selection of specific layers, freezing them and dropping layers that are not used for our specific purpose.

4.4 Reinforcement Learning

Reinforcement Learning (RL) is a powerful machine learning paradigm inspired by how humans and animals learn by interacting with their surroundings. It is a branch of artificial intelligence in which an agent learns to make a series of decisions in a given environment in order to maximize a cumulative reward. RL is especially well-suited to challenges in which explicit supervision is difficult or impossible, and the agent must explore and learn from its own behaviors.

Using RL (Reinforcement Learning) in a math word problem equation generator, helps the model to generalize the problem. T5, a pre-trained language model, is particularly adept at comprehending and producing textual material, including mathematical formulae. Accurate equation generation, on the other hand, involves more than just language understanding, it requires context-aware decision-making. This gap is filled by RL, which provides a framework for the model to learn and optimize its equation generation policy based on feedback. With RL, the system can learn from the validity of the equations it generates, increasing its ability to produce exact and contextually appropriate math solutions. This combination of language understanding and taught decision-making via RL improves the equation generator's overall accuracy and reliability, making it a tool for solving arithmetic word problems successfully.

4.4.1 Core Components of Reinforcement Learning

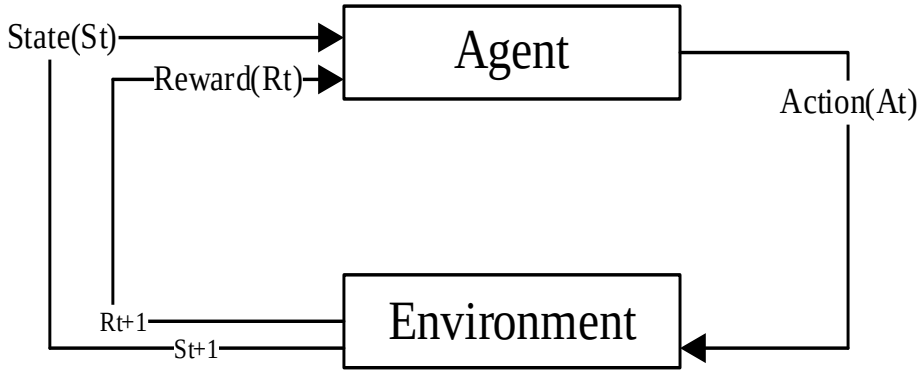


Figure 3- 3: Agent Environment Interaction

- **Agent:** The primary decision-maker in our system is the agent. It interacts with its surroundings, takes actions, and learns from the outcomes. The T5 model is the agent in our scenario. T5 has been taught to comprehend and generate text, including mathematical equations.
- **Environment:** The external context or system in which the agent acts is represented by the environment. The environment includes: Math word problems (input descriptions) for the math word problem solver.
- **States:** States define the agent's current state or context within the environment. States are the input math word problems given to the agent. Text descriptions are used to represent these issues.
- **Actions:** The agent's actions are the decisions or steps it takes to impact the environment. For our system, actions are the mathematical equations generated by the T5 model in response to the input problems.
- **Rewards:** Rewards are numerical numbers that serve as feedback to the agent, showing how effective or ineffective its activities are in the given situation. Rewards in application are linked to the validity of the equations generated by T5. The agent's learning process is guided by rewards, which encourage it to develop accurate equations.

4.4.2 Markov Decision Processes (MDPs)

MDPs are a mathematical framework for modeling decision-making in dynamic and sequential situations. In an MDP, an agent interacts with its environment in discrete time steps. In the context of MDPs, transition probabilities define the likelihood or potential of moving from one state to another once a given action is taken.

A State S_t Markov if and only if

$$P[S_{t+1}|\, S_t] = P\left[S_{t+1}|\, S_{1,\ldots,}S_t\right] \quad 4.7$$

The state captures all relevant information from the history. Once the state is known, the history may be thrown away. i.e. The state is a sufficient statistic of the future.

4.4.3 The Bellman Equation:

The Bellman Equation expresses the optimal value of a decision problem recursively. It decomposes the problem into smaller subproblems and facilitates the computation of optimal policies. It can be expressed in two forms according to state values and for action values:

4.4.3.1 For State Values

The optimal value of being in a state s is equal to the maximum expected return that can be achieved from that state by following the optimal policy π^*. Mathematically it is represented as:

$$V^*(s) = max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V^*(s')] \quad 4.8$$

Where:

- $V^*(s)$ is the optimal value function for state s.
- a represents possible actions in state s.
- $p(s', r|s, a)$ is the transition probability of moving to state s' and receiving reward r given that action a is taken in state s.
- γ is the discount factor, representing the importance of future rewards.
- $V^*(s')$ is the value function for the next state s'.

- max denotes taking the maximum over all possible action

4.4.3.2 For Action Values

The optimal value of taking an action a in state s is equal to the maximum expected return that can be achieved by taking that action in that state and then following the optimal policy thereafter. Mathematically, it can be represented as:

$$Q^*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma max_{a'} V^*(s')] \quad 4.9$$

Where:

- $Q^*(s,a)$ is the optimal Q-value function for taking action a in state s.
- a' represents possible actions in the next state s'.
- $max_{a'}$ denotes taking the maximum over all possible actions in the next state.

4.4.4 Policies

The policy defines the agent's approach or conduct. It outlines the activities that the agent should do in various conditions. The RL training procedure has an impact on agent's policies. It learns to optimize its equation generating approach in order to maximize the cumulative rewards.

4.4.5 Value-Based Algorithms

Value-based algorithms focus on learning the value function. The expected cumulative reward that an agent can obtain, starting from a particular state and acting according to a particular policy. The value function is used to select the best action to take in each state. There are two main types of value functions:

State-Value Function (V): It gives the expected reward for a state under a particular policy, without specifying the action to take.

Action-Value Function (Q): It gives the expected reward for taking a particular action in a given state and then following a specific policy.

4.4.6 Policy-Based Algorithms

Policy-based algorithms explicitly learn the policy that maps states to actions. The policy can be deterministic, where a state always leads to the same action, or stochastic, where actions are drawn according to a probability distribution.

4.4.7 Proximal Policy Optimization

PPO is a reinforcement learning algorithm used to improve the stability and efficiency of training agents in complex environments. It addresses the difficulty of updating policies while avoiding significant changes, resulting in smoother learning and improved convergence. Policy changes are treated carefully with PPO. A ratio calculation is used to analyze the degree of change in the current policy relative to the prior one. The resulting ratio is then trimmed, preventing the current policy from diverging too much from its predecessor. This clipping process removes the need for the current policy to diverge much from the previous one, hence the term "proximal policy." In essence, PPO promotes stability in the learning process by ensuring a cautious and controlled approach to policy modifications.

The general form of the PPO objective function:

$$L(\theta) = \widehat{E_t}\left[\min\left(\frac{\Pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t, \text{clip}\left(1-\epsilon, 1+\epsilon, \frac{\Pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\right)A_t\right)\right] \quad 4.10$$

Where,

- $L(\theta)$ is the surrogate objective function.
- θ is the parameter vector of the policy network being trained.
- $\Pi_\theta(a_t|s_t)$ is the probability of taking action a_t given state s_t under the current policy.
- $\pi_{\theta_{old}}(a_t|s_t)$ is the probability of taking action a_t given state s_t under the old policy (policy at the previous iteration).
- A_t is the advantage function, representing the advantage of taking action a_t in state s_t over the expected value of taking that action under the current policy.

The term $\text{clip}\left(1-\epsilon, 1+\epsilon, \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\right)$ is the clipping term that ensures the ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ stays within the range [1-ϵ, $1+\epsilon$]. The hyperparameter ϵ controls the size of the trust region.

4.4.8 KL-Divergence

The Kullback-Leibler (KL) divergence stands as a critical component within reinforcement learning algorithms, notably Proximal Policy Optimization (PPO). Within PPO, KL divergence serves as a metric for assessing the difference between the updated policy distribution and the original policy distribution. This divergence measure is fundamental in maintaining controlled and stable policy updates throughout the training process, thereby preventing substantial deviations that may disrupt learning dynamics, ultimately fostering convergence and optimizing policy effectiveness. Particularly in domains such as natural language processing tasks employing transformer models, characterized by intricate and high-dimensional action spaces, the incorporation of KL divergence assumes heightened importance. It facilitates the efficient regulation of the learning process within PPO, ensuring that policy updates strike a balance between being advantageous and conservative. For two probability distributions P(x) and Q(x), the KL divergence from Q to P is defined as:

$$D_{KL}(P\|Q) = \sum_{x\epsilon X} P(x)\log\left(\frac{P(x)}{Q(x)}\right) \quad 4.11$$

Where, D_{KL} (P‖Q) is the KL divergence from Q to P. P(x) and Q(x) are the probability mass functions (or probability density functions for continuous distributions) of the distributions. The sum or integral is taken over all possible values of x in the sample space X.

4.4.9 Gradient Ascent

In RL, the objective is to maximize the expected return over time. Gradient ascent achieves this by iteratively adjusting the parameters of the policy or value function in the direction that increases the expected return. The process of gradient ascent involves computing the gradients of the objective function with respect to the model's

parameters. These gradients represent the rate of change of the objective function with respect to each parameter and indicate the direction in which the objective function increases most rapidly. Once the gradients are computed, they are used to update the parameters of the policy or value function. By repeatedly computing gradients and updating parameters, gradient ascent iteratively guides the agent towards policies or value functions that lead to higher expected returns in the environment.

The equation for gradient Ascent is given by

$$\theta_j = \theta_j + \alpha * \nabla\theta_j J(\theta) \quad 4.12$$

Where, θj is the jth parameter, α is the learning rate and J(θ) is the cost function. The gradient of the cost function, ∇θj J(θ), gives the direction of the steepest ascent, and the learning rate α determines the size of the update step.

4.4.10 Gradient Descent

Gradient descent is an iterative optimization algorithm that is used to find the minimum of a function. It is a first-order optimization method, which means that it uses the derivative of the function to determine the direction of the descent. The algorithm is designed to iteratively update a set of parameters until it converges to the minimum of the function. The gradient gives the direction of the steepest descent, and the magnitude of the update step is determined by a learning rate. The algorithm is then repeated until the parameters converge to a minimum of the function.

The equation for gradient descent is given by

$$\theta_j = \theta_j - \alpha * \nabla\theta_j J(\theta) \quad 4.13$$

The gradient descent is similar to the one for gradient ascent only, with a opposite sign.

4.4.11 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an optimization algorithm for minimizing a loss function, over a parameter space. SGD computes the gradient using a random subset of the training data, providing an estimate of the true gradient making SGD more

computationally efficient and well-suited for large-scale optimization problems. SGD approximates the gradient of the loss function by taking a single sample from the dataset and computing the gradient with respect to the parameters. The algorithm iteratively updates the parameters in the direction of the negative estimated gradient.

Mathematically, consider a loss function L(θ), where θ represents the parameters of the model. The goal is to find the parameters θ∗ that minimize the loss function. The gradient of the loss function with respect to θ using a single data point (xi, yi) from the dataset is computed as:

$$\nabla\theta L(\theta;\ x_i, y_i) \tag{4.14}$$

The iteration update rule for SGD is given by:

$$\theta\ =\ \theta\ -\ \eta * \nabla\theta L(\theta;\ x_i, y_i) \tag{4.15}$$

Where, η is the learning rate, which determines the step size in the parameter space.

4.4.12 Momentum

Momentum is an optimization technique used in conjunction with Stochastic Gradient Descent (SGD) to improve convergence properties and reduce oscillations. SGD algorithm takes larger steps in the direction of the gradient, effectively speeding up the learning process. The momentum method maintains a running average of the gradients, which is used to update the model parameters.

Momentum considers the past gradients and their contribution to the current gradient which identifies the dominant direction of the gradient and taking larger steps in that direction. It reduces the impact of gradients in the opposite direction, effectively "dampening" the oscillations.

For momentum, running average of gradients is updated at each iteration using the equation:

$$V_{(t)}\ =\ \gamma * V_{(t-1)}\ +\ \alpha * \nabla J(\theta) \tag{4.16}$$

Where,

- V(t) is the running average of the gradients at iteration t.
- γ (gamma) is the momentum hyperparameter
- V(t-1) is the running average of the gradients at the previous iteration.
- α∇J(θ) is the product of the learning rate α and the gradient of the loss function ∇J(θ) at iteration t.

The update rule for the model parameters is then given by:

$$\theta = \theta - V_{(t)} \quad 4.17$$

By using this update rule, the algorithm takes larger steps in the dominant direction and reduces the impact of noise and oscillations in the gradient.

4.4.13 Adagrad

Adagrad, which stands for Adaptive Gradient Algorithm, is an optimization algorithm that adapts the learning rate for each parameter based on the past gradients and accumulated sum of squares of gradients. The algorithm to automatically adjusts the learning rate for different parameters.

Adagrad maintains a running sum of the squared gradients, which is used to scale the gradients based on their historical importance. Parameters with larger gradients in the past receive a smaller learning rate, while parameters with smaller gradients receive a larger learning rate which balances the updates for parameters with different scales.

The Adagrad algorithm can be described using the following update rules:

i. Accumulate the sum of squared gradients:

$$Sum\_squared\_gradients = Sum\_squared\_gradients + \alpha^2 * \nabla J(\theta) \quad 4.18$$

Where,

- Sum_squared_gradients is the running sum of squared gradients.
- α is the learning rate.
- ∇J(θ) is the gradient of the loss function with respect to the parameters θ.

ii. Update the parameters using the scaled gradient:

$$\theta = \theta - \left(\frac{\alpha}{\sqrt{(Sum\_squared\_gradients + \varepsilon)}}\right) * \nabla J(\theta) \quad 4.19$$

Where,

ε is a small constant added to the denominator to prevent division by zero.

The learning rate for each parameter is given by (α / sqrt(Sum_squared_gradients + ε)). As the Sum_squared_gradients increase over time, the learning rate decreases preventing the explosion of parameter updates.

4.5 Adam Optimizer

The AdamW optimizer is designed to adapt the learning rate for each parameter individually, based on the magnitude of the gradient. The AdamW optimizer works by maintaining two separate moving averages of the gradient and its square, and using these values to update the parameters of the model. The update rule for the parameter wi at time t is given by:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}}{\sqrt{\hat{v}_t}+1} \quad 4.20$$

Where,

- θ_{t+1} is the parameter at time t
- θ_t is the parameter at time t-1
- η is the learning rate
- $\hat{m}_t$ is the moving average of the gradient
- $\hat{v}_t$ is the moving average of the gradient-squared
- ε is a small value added for numerical stability

The moving averages $\hat{m}_t$ and $\hat{v}_t$ are updated as follows:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad 4.21$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad 4.22$$

Where,

- g_t is the gradient at time t
- β is the beta1 hyperparameter, which controls the decay rate of the moving averages

The AdamW optimizer also includes a modification to the learning rate, which is defined as:

$$\eta = \eta_0 \times (1 - \beta) \times (1 + \gamma) \quad 4.23$$

Where,

- η_0 is the initial learning rate
- β is the beta1 hyperparameter
- γ is the beta2 hyperparameter, which controls the decay rate of the learning rate.

4.6 Hyperparameter for RL Application

4.6.1 Top-K sampling

Top-k sampling is a probabilistic sampling method used during text generation to limit the set of tokens considered for sampling at each step. In this method, only the top-k most probable tokens according to their predicted probabilities are retained, where k is a hyperparameter specifying the size of the set. The probabilities of the remaining tokens are normalized, and one token is then randomly sampled from this reduced set based on their probabilities. Top-k sampling encourages the model to explore diverse vocabulary while still ensuring that the generated text remains fluent and coherent.

4.6.2 Top-P sampling

Top-p sampling, also known as nucleus sampling, is another probabilistic sampling technique used during text generation. Unlike top-k sampling, which considers a fixed number of tokens, top-p sampling dynamically selects tokens based on their cumulative probability mass. In this method, tokens are selected until the cumulative probability mass exceeds a threshold parameter p, which determines the nucleus of the probability distribution. This threshold is adaptive and allows for a variable number of tokens to be considered for sampling at each step. Top-p sampling is effective in promoting diversity in generated text while ensuring that the most probable tokens still receive preference.

4.7 System Block Diagram

The proposed methodology is a promising approach for developing a Linear Equations math word problem solver. By combining T5 and policy-based RL, a Linear Equations word problem solver is developed that is able to generate accurate and efficient solutions to a wide range of problems.

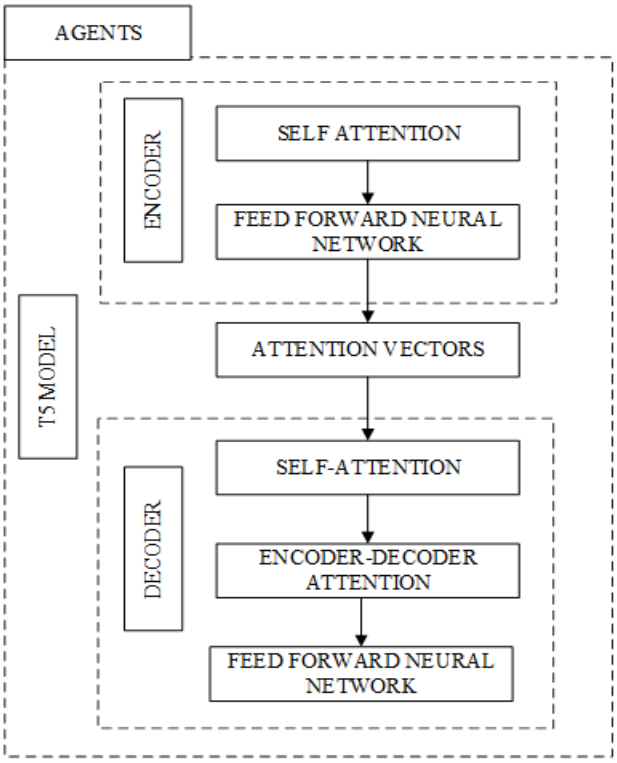


Figure 3- 4: T5 Model

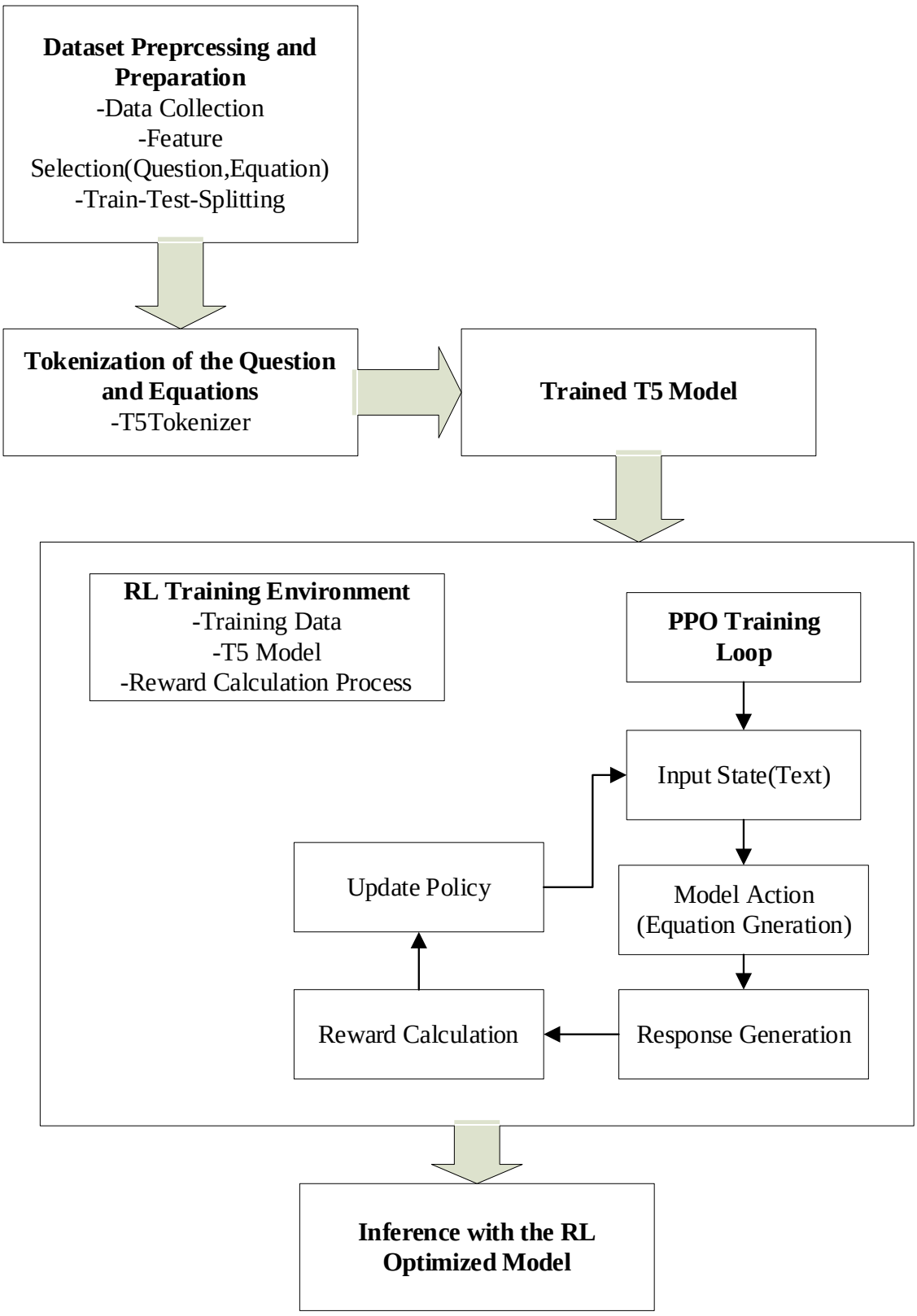


Figure 3- 5: System Block Diagram of T5 and policy based RLs

4.8 Class Diagram

The Class Diagram represents the system designed to generate math equations from math word problems using a large language model i.e. T5 model and Reinforcement Learning. Following are the key components and their interaction.

Classes of Equation Generator class diagram

- **Dataset:** This class represents the data used to train and evaluate the model. It stores text data in the form of paired word problems and their corresponding equations.

- **MathWordProblemEquationGenerator:** This is the main class responsible for generating equations for a given word problem. It interacts with other classes like T5Model and EquationGenerator.

- **T5TrainingPipeline :** This class represents how the model is trained to generate equations from math word problems.

- **T5Model:** This class represents the pre-trained T5 language model, fine-tuned on the provided dataset for generating equations from word problems. It takes the word problem text as input and generates equations.

- **RLEnv:** This class represents the environment the reinforcement learning agent interacts with. It evaluates the agent's actions based on the word problem and provides a reward signal based on the correctness of the generated equation.

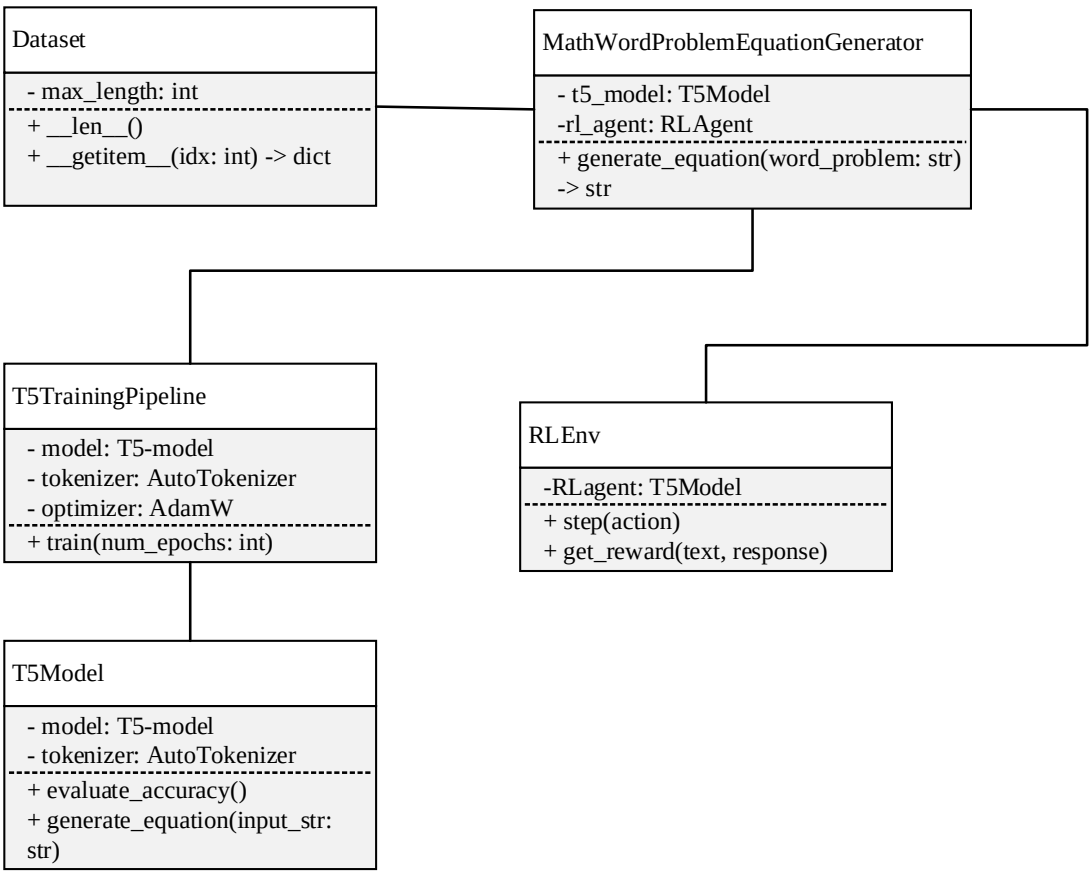


Figure 3- 6: Class Diagram

4.9 Use Case Diagram

The use case diagram is a visual representation of the interactions between a system and its users. It depicts the functionalities of a system from the perspective of its users, focusing on what the system does rather than how it does it. It depicts the system designed to generate equations from textual math word problems.

- **User Entity**

  Represents individuals requesting automated assistance in turning word problems into equations. he user provides a textual description of a math problem as input, expecting the system to interpret and translate it into an equation. The system responds with generating equations corresponding to the provided word problem.

- **System Admin Entity**

  Represents individual responsible for the maintenance of the system. The system allows adding new word-problem-equation pairs to the system's dataset, potentially

enriching its understanding and generating capabilities. The existing dataset is used to train or fine-tune the underlying model, aiming to enhance its accuracy, efficiency, and generalizability in generating equations.

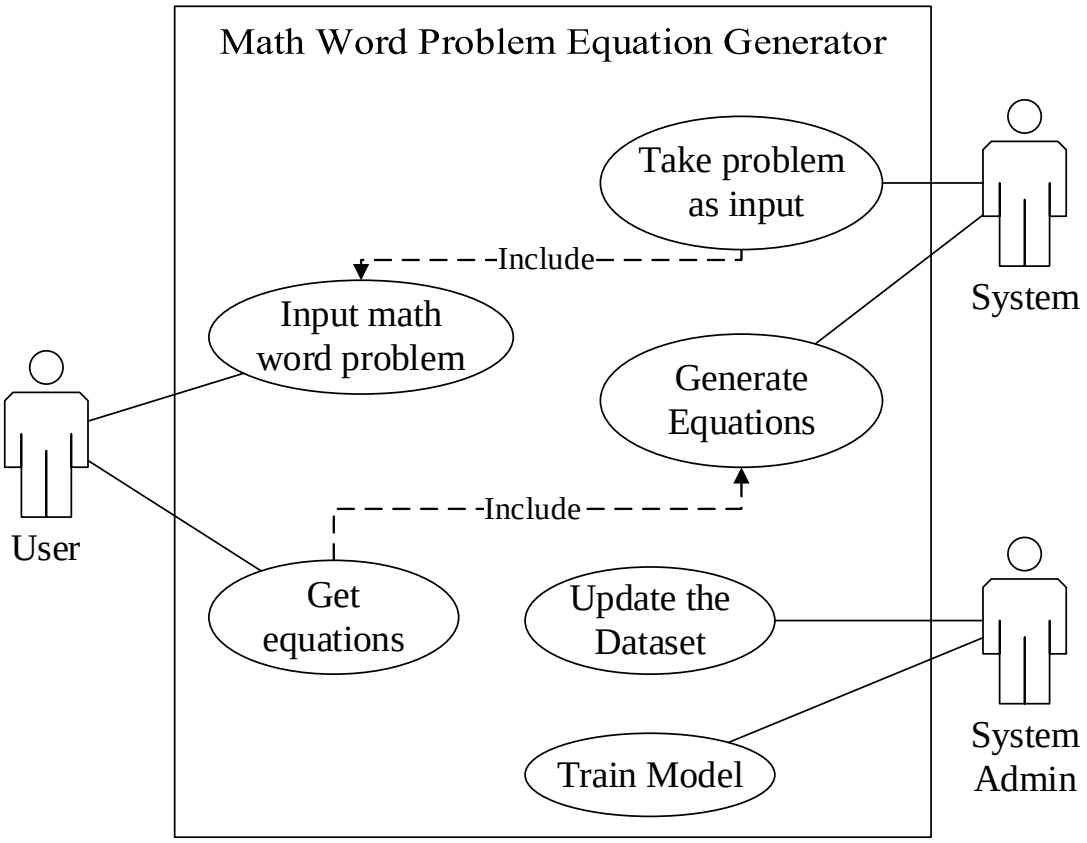


Figure 3- 7: Use Case Diagram

4.10 Activity Diagram

The activity diagram of Linear math word problem equation generator shows the flows between the activity of dataset, T5 equation generation, RL environment, PPO Training as RL agent to training loop and equation generation with RL agent. The diagram provides a comprehensive overview of the process involved in the math word problem equation generator using T5 and policy-based RL, making it easier to understand the entire workflow.

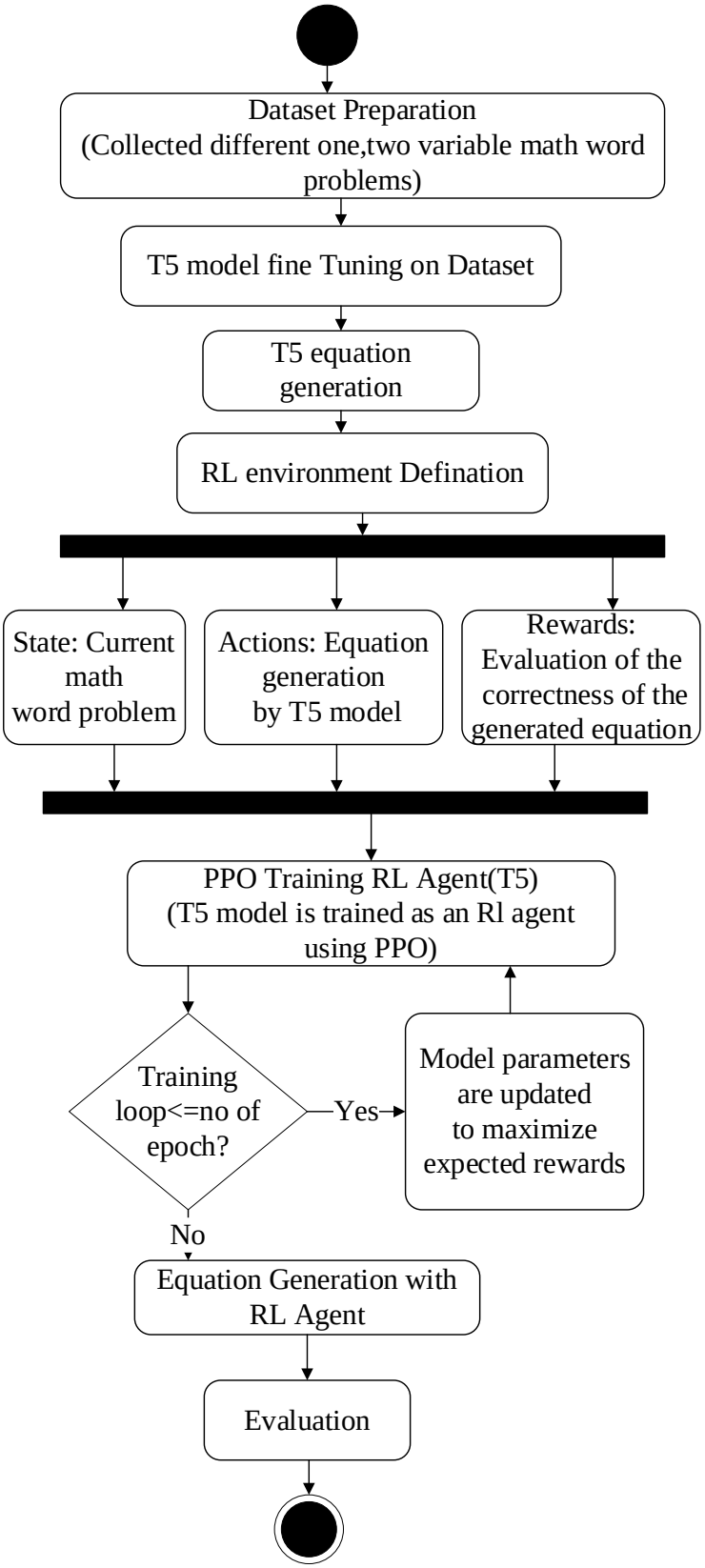


Figure 3- 8: Activity Diagram

4.11 State Diagram

State Diagram incorporated the user interface interactions for the Linear math word problem equation generation. User enters a math word problem through the UI. The system processes the word problem using the T5 model and generates an equation. The generated equation is displayed to the user via the UI. This state diagram captures the basic flow of interactions between the user and the system in generating and displaying equations for math word problems. It focuses on the key states and transitions relevant to the UI interactions.

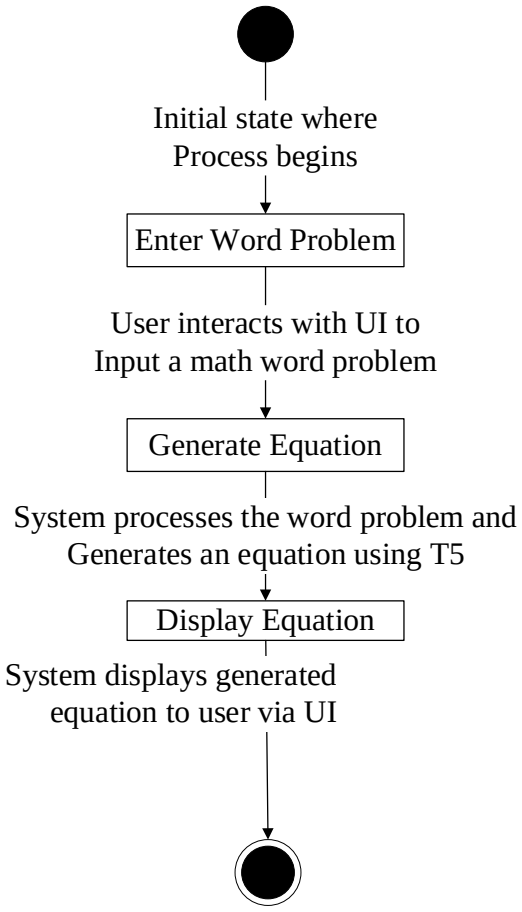


Figure 3- 9: State Diagram

4.12 ER Diagram

This ER (Entity Relationship) Diagram represents the model of Linear math word Problem Equation Generator. The entity relationship diagram of this system shows all the visual instrument of database tables and the relations between User, Word Problem, Generated Equation. It is used to structure data and define the relationships between

system structured data groups. The main entities of the system are Word problem, Generated Equation, User.

- Word Problem: This entity represents the math word problems input by users.
  Attributes: problem_id: Primary Key (PK) uniquely identifying each problem, text: The text of the math word problem.

- Generated Equation: This entity stores the equations generated by the system for each word problem.
  Attributes: equation_id: Primary Key (PK) uniquely identifying each generated equation, problem_id: Foreign Key (FK) referencing the problem_id in the Word Problem entity, equation_text: The text of the generated equation.

- User: This entity represents the users of the system.
  Attributes: user_id: Primary Key (PK) uniquely identifying each user, username: The username chosen by the user.

- Model: This entity represents the machine learning model component used for equation generation i.e T5 Model and RL
  Attributes: model_id: Primary Key (PK) uniquely identifying each model, model_name: The name of the model

Explanation of relationships:

- Word Problem → Generated Equation:
  Decision: One-to-Many
  Explanation: Each word problem can have multiple generated equations (one-to-many relationship) since different models or iterations of the same model may produce different equations for the same problem.

- Word Problem → User:
  Decision: Many-to-One
  Explanation: Multiple word problems can be submitted by the same user (many-to-one relationship), as a single user can input multiple math word problems.

- Model → Generated Equation:

  Decision: One-to-Many

  Explanation: Each model can produce multiple generated equations (one-to-many relationship), as it can be used to generate equations for multiple word problems.

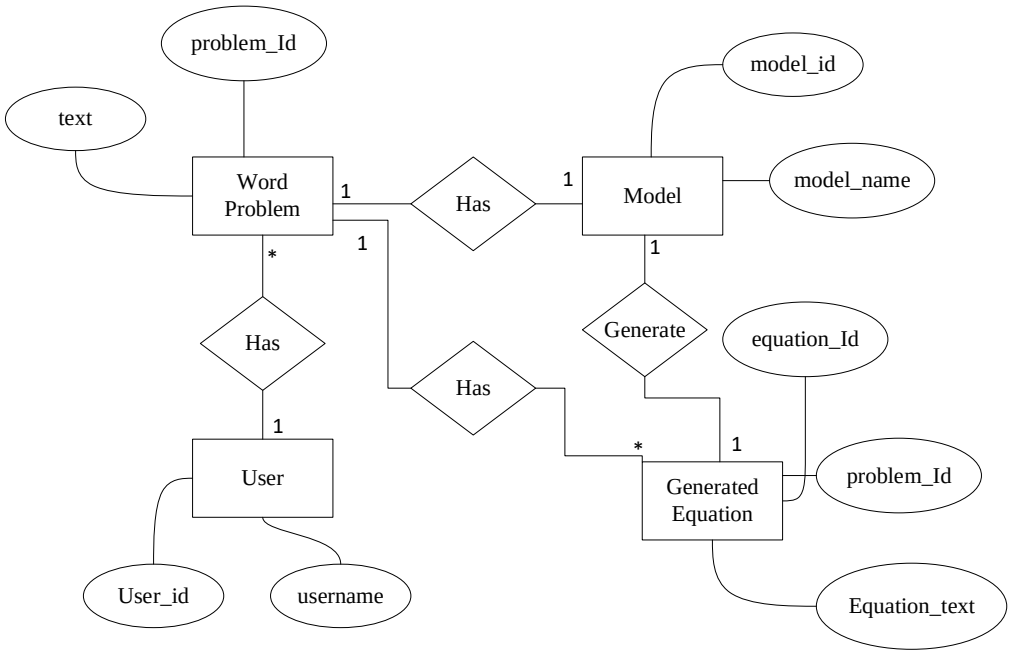


Figure 3- 10: ER diagram for Linear Math Word Problem Equation generator

4.13 Data Flow Diagram

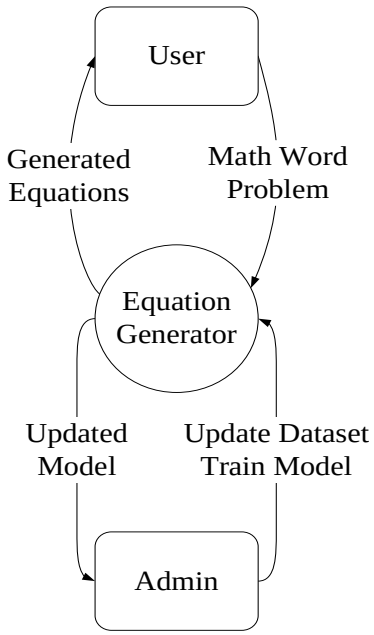


Figure 3- 11: Data Flow Diagram

The equation generator system consists of three primary components: The Equation Generator, the User, and the Administrator. The Equation Generator is the main component that generates equations from input math problems. Users interact with the system by entering mathematical questions and receiving equations generated by the Equation Generator. Administrators, on the other hand, enjoy special access and obligations. They update the dataset utilized by the Equation Generator and train the model using the revised data. Additionally, administrators receive the revised model. The User-Generator interaction involves users inputting math problems and obtaining corresponding equations, while the Admin-Generator interaction entails dataset updates and model training. This contextual data flow diagram illustrates the high-level interactions between these components, focusing on user inputs, equation generation, dataset updates, and model training. It provides a clear overview of how different entities interact with the system without delving into internal processes or data flows within each component.

4.14 Dataset Preparation

4.14.1 Dataset Compilation

The data collection process involved obtaining datasets from various sources. Due to limited number of datasets many of the one and two variables' questions were prepared. Each source provided different question and answer datasets related to mathematical problem-solving. Datasets were obtained in diverse formats, making the initial data compilation challenging.

Table 3- 1: Sources of Dataset

Dataset	Question	Equation	Answer	Number of datasets
dolphin1878	What is the number x in 2+x/7 = 20/7?	['unkn: x', 'equ: 2+x/7 = 20/7']	[6]	1878
DRAW	A bank teller notices that he has 50 coins all of which are 5c and 10c pieces. He finds that the value of the coins is $ 4.20. How many of each must he have?	0.05*m+0.1* n=4.2; m+n=50.0	[16.0, 34.0]	996
MAWPS	You are reading a book with 120 pages. If you want to read the same number of pages each night, how many would you have to read each night to finish in 10 days ?	X=(120.0/10. 0)	12	2222
number_word	One number is 3 less than a second number. Twice the second	['unkn: x,y', 'equ: x + 3 =	[6, 9]	4372

	number is 12 less than 5 times the first. Find the two numbers.	y', 'equ: 2*y + 12 = 5*x']		
Prepared Dataset	6 ducklings in the first swim, 8 ducklings in the second swim, how many ducklings in the second swim.	x=6+8	X=14	22000

4.14.2 Dataset Pre-Processing

Data preprocessing is a crucial step in the data analysis pipeline that involves cleaning, transforming, and organizing raw data into a format suitable for further analysis, modeling, and machine learning tasks. In the context of the collected dataset for mathematical problem-solving, the data preprocessing phase involved several detailed steps to ensure the dataset's quality, consistency, and relevance.

- **Cleaning Dataset**

  The cleaning process addressed various data quality issues within each dataset to ensure reliable and accurate information for analysis. Missing values were handled by removing records with missing data. Duplicate entries were identified and eliminated to prevent redundancy and ensure each question-equation pair is unique.

- **Feature Selection - Question, Equation**

  To standardize the dataset format, each dataset was transformed into a consistent structure with three two columns 'question,' 'equation,'. The 'question' column contained the problem statements in a human-readable and standardized format. The 'equation' column captured any relevant mathematical equations associated with each question, facilitating further mathematical analysis and modeling.

- **Combining the Collected Datasets**

  After cleaning and restructuring each dataset, they were combined into a single,

unified dataset to create a comprehensive and diverse data repository. The merging of datasets from different sources aimed to create a larger and more representative dataset suitable for robust model training and evaluation.

- **Tokenization**

  The tokenizer plays a crucial role in preprocessing the input text data before feeding it into the model. The tokenizer is responsible for breaking down the input text into individual tokens, which are the basic units of text processing in the model. The tokens can be easily processed and analyzed by the T5 model. The tokenizer adds special tokens to separate starting and end on an equation. The tokenizer generates attention mask which is a binary vector which indicates what tokens the model should pay attention to and which tokens to ignore. Each token are mapped to their individual IDs based on the vocabulary.

4.15 T5 and RL Algorithm: Math Word Problem Equation Generator

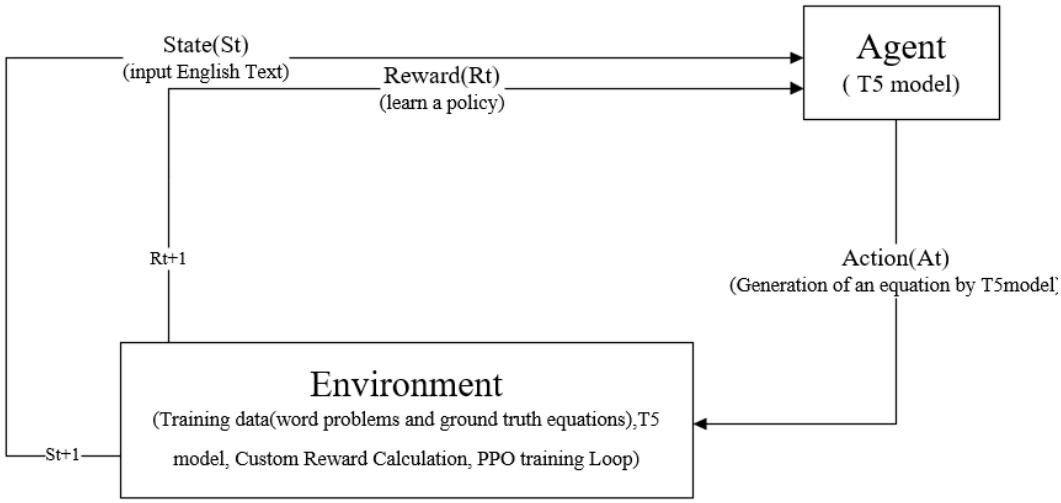


Figure 3- 12: Math word problem Agent Environment Interaction

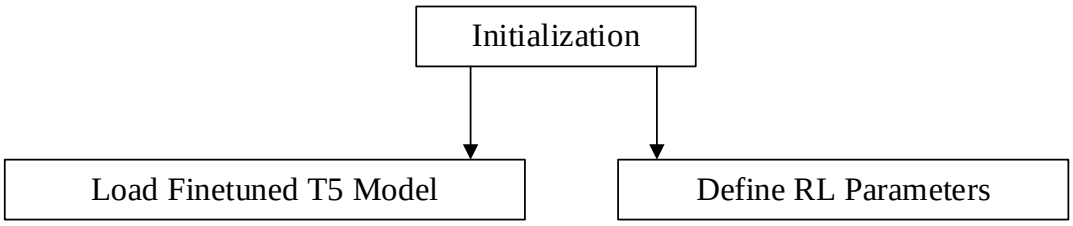


Figure 3- 13: Initialization

Initialization: Load the fine-tuned T5 model trained to generate math equation to a given word problem parameters such as environment, action and reward function are defined.

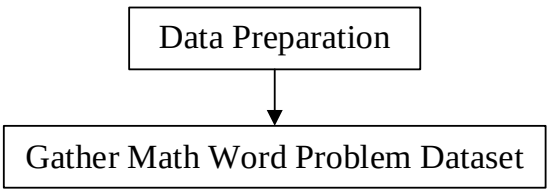


Figure 3- 14: Data preparation

Preparation of Data: Arithmetic word problems, together with the proper equations is defined. These word problems are used to train the RL agent.

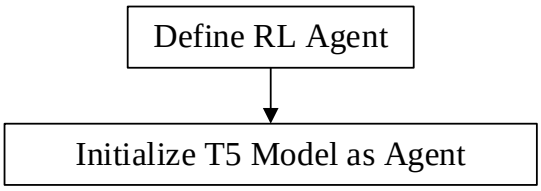


Figure 3- 15: Agent defining

Define the term RL Agent: The T5 model is the RL agent. It is used to produce equations for given math word problem.

Define Environment: The environment represents the set of all math word problem. The agent interacts with the environment to generate corresponding equation.

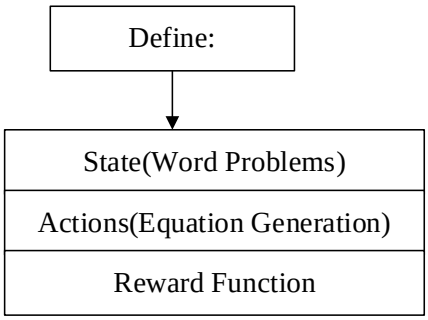


Figure 3- 16: Defining State, Action, Reward

Define States: The states are the input math word problems presented to the T5 agent. Each state is a distinct arithmetic word problem. The agent performs different action based on the given state. The agent performs action that maximizes the reward.

Define Actions: The T5 model generates equations through actions. The model predicts sequence of tokens (i.e. equation) for a given problem.

Define the Rewards: The correctness of the generated equations is compared against the right equations in the dataset to determine rewards. Reward is assigned between 0 and 1 according to the quality of equation generated.

Policy Definition: The policy specifies how the T5 agent creates equations from the current state (math word problem). Policy is the brain of the agent. It is basically a function that maps the state with best corresponding action. Policy is optimized using Proximal Policy Optimization which is a policy-based RL algorithm.

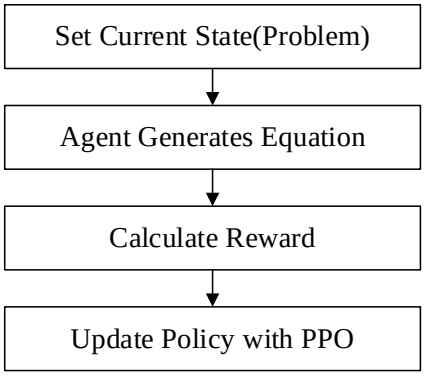


Figure 3- 17: RL Training

R L Training Loop

Perform RL training to fine-tune the T5 agent for generating accurate equations.

- For each problem in the dataset
- Set the current state as math word problem.
- The agent (T5 model) chooses an action by generating an equation.
- Th reward is calculated based on the correctness of the generated equation compared to the correct equation.
- Update the policy of the T5 agent using a Proximal Policy Optimization algorithm.

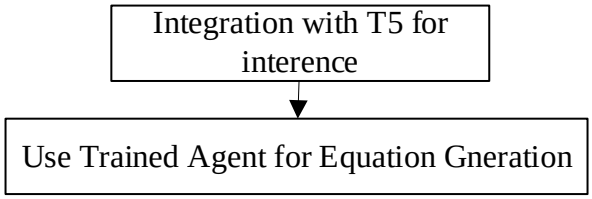


Figure 3- 18: Integration with T5

4.16 Equation Solver

Leveraging the Math Steps JavaScript Node package, primarily designed for single-variable problems, an equation solver capable of handling up to two variables was developed. To address two variable complexities, a standardized modification process was implemented. This process involved transforming one-variable equations into a specific format and subsequently substituting their solutions into the second equation. By effectively converting both equations to single-variable forms, Mathsteps could be harnessed to efficiently solve the system.

The adopted approach, outlined in the following steps, facilitates the solution of such equations:

1. **Arithmetic Operator Inclusion:** In the absence of a preceding arithmetic operator before any operand (variable), one is introduced for enhanced clarity. Example**:** x + y = 2 becomes +x + y = +2.

2. **Equation Segmentation:** The equation is divided after the '=' operator. Example**:** +x = +2 + y
   Left side: +x
   Right side: +2 + y

3. **Operator Modification:**
   a. **No Parentheses:** If no parentheses are present on the right side, arithmetic operations are reversed (plus to minus, and vice versa). Example**:** Right_side = +2 + y is changed to Right_side = -2 - y.

   b. **Parentheses Present:** Only the operators outside the parentheses are modified.
   Example**:** Right_side = +2 + (x + 1) becomes Right_side = -2 - (x + 1).

4. **Equation Standardization:** The equation is transformed to the format ax + by + c = 0.
   a. The right-side string is added to the left side.

   b. A zero is added after the '=' operator.
   Example:
      i. Left side: +x
      ii. Right side: -2 - y
      iii. New equation: +x - 2 - y = 0

5. **Variable Position Adjustment:** To ensure x always precedes y, equations in the format ay + bx + c = 0 are restructured to bx + ay + c = 0.

6. **Value Substitution and Solution:** Following standardization, Mathsteps generates the x value in terms of y. This x value is subsequently substituted into the second equation, which is then solved for y.

7. **X Value Determination:** The solved y value is then substituted back into the

first equation to determine the value of x.

4.17 Iterative Model

Iterative Model is one of the approaches for software development in the whole process of software development is divided into smaller portions. It does not accomplish the complete specification within a cycle. It starts satisfying the minimum requirements of the system and iteratively moves onto the complex one. It begins by specifying implementing the part of software. Prototypes are reviewed and analyzed to add further requirements which form the iteration of development.

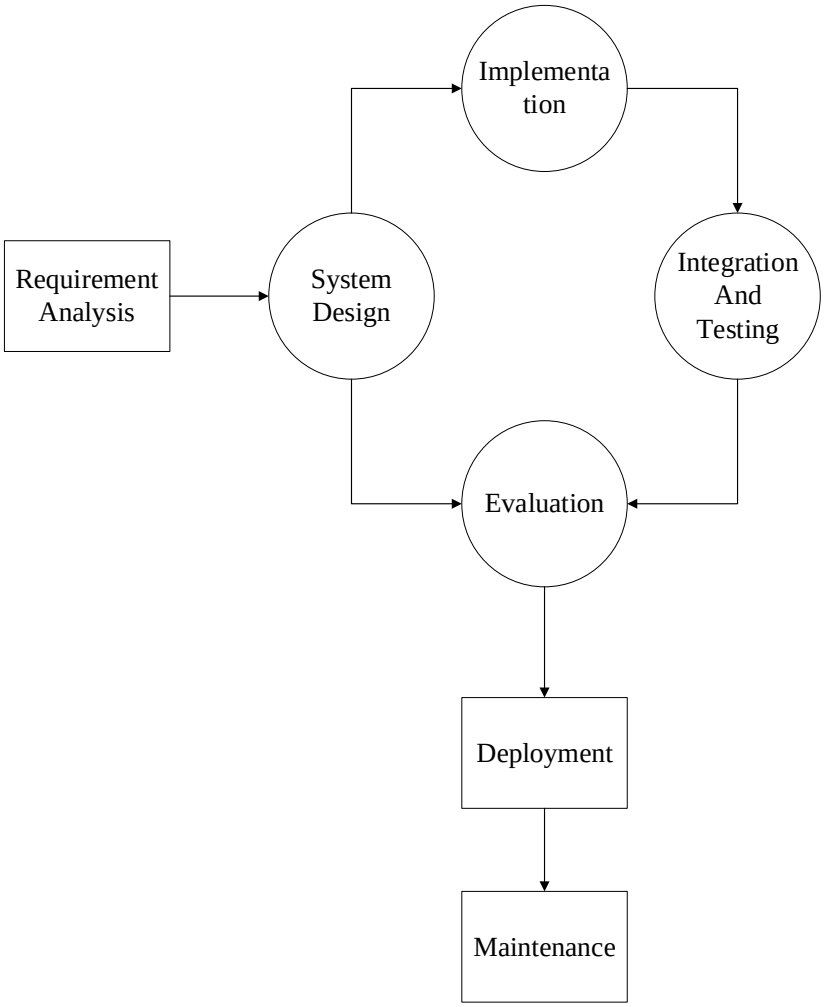


Figure 3- 19: Iterative Model

Justification of applying the Iterative Model along with description of its phases are:

4.17.1 Requirement Analysis

In the initial phase of requirement analysis, the team explored the fundamental needs of the equation generation project. This involves understanding the dataset's characteristics, defining desired equation formats, and structuring potential reward mechanisms for reinforcement learning (RL). Valuable insights from literature reviews and research papers contributed to refine and expand evolving requirements.

4.17.2 System Design

During system design, an initial high-level plan is developed, outlining the architecture, components, and interactions among RL, transformer models, and the dataset. This design is flexible, allowing for adaptation as requirements evolve. Ongoing collaboration ensures alignment with user expectations. It is emphasized that the model is designed to handle questions within the dataset and may not extend to solving questions outside its known scope.

4.17.3 Implementation

The implementation phase commences with the creation of a baseline version, concentrating on core functionalities like RL training and transformer integration. Features are added incrementally, addressing specific requirements and challenges identified earlier. The model's capability is confined to solving questions present in the dataset, managing user expectations and providing clarity on the system's limitations.

4.17.4 Integration and Testing

Component integration proceeded incrementally to ensure seamless interaction between RL and transformer components. Continuous testing is integral at each integration step, with an awareness that the model's effectiveness is contingent on the dataset's representativeness. Any identified issues are promptly addressed, aligning with the iterative and adaptive nature of development.

4.17.5 Evaluation

The evaluation phase assesses the initial equation generation capabilities against a small dataset subset. Feedback from users or simulated environments informs adjustments to

models and algorithms. Also, continuous the dataset was increased consisting of various kinds and again moved for training.

4.17.6 Deployment

Partial deployment follows, releasing the system in a controlled environment for partial usage and user feedback. Continuous deployment strategies are employed for iterative releases, ensuring user feedback is promptly incorporated.

4.17.7 Maintenance

In the maintenance phase, continuous monitoring identifies and addresses issues promptly. Adaptive maintenance strategies accommodate changes in dataset characteristics or evolving educational requirements. The team emphasizes the model's limitation in handling questions within the dataset, maintaining transparency about the system's capabilities and boundaries.

5 IMPLEMENATION DETAILS

In this project, Math equation generator from word problems, T5 pretrained base model is used which is provided by Hugging Face's Transformers library. The T5 transformer's capabilities is utilized to effectively transform natural language math word problems into corresponding mathematical expressions. To fine tune this model, large dataset consisting of the questions and equations is required. The major steps involved in this project are as following:

- Dataset Collection
- Dataset preprocessing
- Tokenization
- Training
- Evaluation

5.1 Dataset collection

Dataset are collected from the various sources. Diverse set of math word problems covering various topics, level of difficulty and styles are available. Dataset of math word problems consisting of one or two variables and linear equations were given priority. Collected dataset were of different formats and styles.

5.2 Dataset preprocessing

It is the most essential phase before training the model. This involves cleaning, filtering of the dataset. From the collection of datasets, dataset is filtered on the basis of scope of problem, number of variables. Then dataset is cleaned to that format that T5 will accepts.

5.3 Tokenization

For the tokenization process, T5 tokenizer has been used which is provided by transformers library. Transformers library provides us T5Tokenizer class and the instance of the T5 Tokenizer class is created by calling a class method from_pretrained(). This method loads the pre-trained tokenizer configuration and vocabulary associated with t5-model and the instance is assigned to tokenizer variable.

As T5 is already pre-trained model and Its performance can be enhanced by prepending a different prefix to input related to the task. So as math word problem is to be converted into equations, so the prefix English to Math Equations are added. After this, the List of input text is passed to the tokenizer object as argument and it also accepts the following argument too

- Inputs: Input text after adding prefix
- Padding: padding=True is set. It is used to ensure that all input sequences have the same length. It adds the special tokens to the end of shorter sequences until they reach the specified maximum length
- Truncation: Truncation=True has been set. It is used to handle the sequence that exceed a predefined a maximum length by removing extra tokens.

Then, tokenizer object returns following:

- Input_ids: Input ids holds the tokens ids of the input text. Tokens IDs are the integer values that represent the individual tokens on the basis of their position in the tokenizer's vocabulary. Each token is assigned a unique Integer IDs.
- Attention_mask: Attention_mask is most crucial input to the model as it tells the model which are actual tokens and padding tokens. So, the model attends to the actual token only. In the padding process, some padding tokens have been added to make all inputs have same length.

5.4 Model Training

The pre-trained model is trained on a specific downstream task using task-specific data. But before actual model training, following tasks are performed.

- Device Selection
- Data Loading
- Setting hyperparameter

Device Selection: torch.device() function which is provided by the PyTorch library , which is used to select the device on which PyTorch tensors and computations will be performed. This function will set the device to the GPU if it is available. GPU is used to perform parallel processing which then leads to the faster computations.

Data Loading: It is use to efficiently organize and process the input data. It makes input data ready for training in PyTorch. It helps streamline the training process while optimizing memory usage and computational efficiency. It consists of two steps:

- Creating the Dataset: A Tensor Dataset is created using the input_ids, attention_mask and labels. Tensor Dataset is a PyTorch class that takes multiple tensor as input and combines into a unified dataset
- Creating Dataloader: Dataloader is created using DataLoader class. Batch size is passed as another parameter to DataLoader class. It provides an iterable over the dataset, allowing the models to process data in batches.

Setting Hyperparameters

To control the training of model, the some of the hyperparameter are defined. They are as following:

- **Batch size (16)**: The batch size determines the number of samples processed together during each training iteration, balancing computation efficiency and model stability.
- **Number of epochs (10)**: The number of epochs defines how many times the entire dataset is passed through the model during training, influencing model convergence and performance.
- **Optimizer (Adam optimizer)**: Adam optimizer is a popular optimization algorithm that adapts learning rates for each parameter individually, enabling efficient training and faster convergence.
- **Learning rate (1e-4)**: The learning rate controls the step size during optimization, affecting the speed and quality of model updates during training.
- **Dropout (0.1)**: Dropout is a regularization technique that randomly drops a fraction of input units during training, reducing overfitting by encouraging robustness and generalization.

5.4.1 Initializing the pre-trained model

T5 model is initialized from the class T5ForConditionalGeneration provided by transformer library. By using from pretrained method provided by this class, model is initialized by passing the model's name parameter (t5-base) to this method.

To start the fine-tuning process of pre-trained model, model is set in training mode, as it enables the operation like dropout regularization. Dropout regularization, certain fraction of neurons in neural network are randomly dropped out which temporarily exclude these neurons from the backward and forward pass. It is performed to force the model to learn redundant representations and distribute the learning across different combinations of neurons.

Initial Parameters

- Total loss=0
- Zero Gradient

After setting initial parameters, the batch of data is passed to the model for forward pass. Data loader is used to iterates over batch of data. Common device is used for training purpose, so input_ids, attention_mask and labels are extracted to common device, in this case common device is GPU. Then the forward pass is performed by calling the model object and passing the input_ids, attention_mask and labels to that object and this returned value is stored to the output variable.

The output object contains the loss attributes which provides the loss value which is scalar quantity which specify how different the model's predictions form the ground truth value. Model uses cross entropy loss function to evaluate the loss value.

Cross-entropy loss is a measure of how well a probability distribution P matches target distribution Q. It is given by the formula

$$\text{CrossEntropyLoss}(P, Q) = -\sum (Q_i * \log(P_i)) \quad 5.1$$

Where,

- Q_i is the probability of the i th element in the target distribution Q.
- P_i is the probability of the i th element in the predicted distribution P.

The cross-entropy loss quantifies the dissimilarity between the predicted probabilities and the true probabilities, represented by the target distribution. The goal during training is to minimize this loss, which encourages the model to improve its predictions to better match the target distribution.

5.4.2 Back-propagation

- Gradient computing: The backward () method is used to compute the gradients of the loss with the respect to the model's parameters. This step is crucial part of the backpropagation algorithm, which allows the model to update its weights based on the computed gradients.
- Optimization: It is used to update the models' parameters. Adam optimizer has been used. Adam optimizer is algorithm that is commonly used to update the model's parameters during the training in deep learning. It is used because of its fast convergence and robustness to different learning rates. It combines elements of both the Adagrad and RMSprop optimizers.

This overall process is repeated for multiple iteration and over multiple epochs. In each iteration, the optimizer makes small adjustments to the weights based on the gradients and learning rate. As the training progresses, the model's weights are iteratively updated to minimize the loss on the training data. This iterative process aims to find the set of weights that best capture the patterns in data and generalizes well to new data.

5.5 Custom Reward Model

The custom reward is generated based on the match of the generated equation with the true equation for a given question. Similarity ratio is calculated between the generated equation and the true equation. Another reward is based on the length of the generated equation and the predicted equation. The length of the generated equation is compared with the original equation, if the length of the predicted equation matches that of the true equation then the agent gets full reward which is one. Else the model gets penalty according to their length ratio between the generated and the true equation. For calculating the similarity ratio firstly, the longest common subsequence of characters that appear in the same order in both of the strings. Then the similarity ratio is calculated by taking the length of the longest common subsequence and dividing it by the length of the two sequence being compared. Final reward is calculated by multiplying the similarity ratio and the length penalty.

$$Similarity\ ratio\ = \frac{L}{max(N,M)} \tag{5.2}$$

Where,

- L is the length of longest common subsequence,
- N and M are the length of two strings being compared.

5.6 Evaluation Metrics

Our implemented math word problem solving system is being evaluated using a complete examination that includes Equation Matching (EM), ROUGE-1, ROUGE-2, ROUGE-L, and a final answer comparison.

i. Equation Matching

To begin, the Equation Matching metric is used to assess the accuracy of the system's equations. EM evaluates how well the generated equations match the reference equations. A high EM score indicates that the system captures the predicted mathematical expressions successfully, indicating its capacity to formulate appropriate equations.

ii. ROUGE

ROUGE is a popular metric for assessing the quality of text generating jobs. It determines how well a generated text fits with a set of reference texts, which are usually human-written versions of the same content. Despite its limitations, ROUGE provides useful insights regarding a model's capacity to capture relevant information, fluency, and stylistic characteristics of language.

- N-grams: ROUGE evaluates the overlap of n-grams between the generated and reference texts. ROUGE variations capture varying levels of granularity by using different n-gram lengths.
- Recall: ROUGE calculates the percentage of n-grams from reference texts that appear in the created text. This emphasizes how much more information the generated text conveys than the references.
- Precision: ROUGE also considers the opposite direction: the percentage of n-grams in the generated text that are also found in the reference texts. This determines how much redundant or irrelevant information the resulting text contains.

Different versions of ROUGE are used to access the accuracy of generated equation from the model.

a. ROUGE-1

ROUGE-1 is a metric for evaluating machine-generated text automatically, particularly in natural language processing and text summarization. It calculates the percentage of unigram (single-word) tokens that overlap between the generated and reference text.

The formula for ROUGE-1 is as follows:

$$\text{ROUGE} - 1 = \frac{\text{Number of overlapping unigrams in generated and reference text}}{\text{Total number of unigrams in reference text}} \quad 5.3$$

Where,

- **Number of overlapping unigrams in generated and reference text:** This is the number of unique words that appear in both the generated and reference text.

- **Total number of unigrams in reference text:** This is the total number of unique words in the reference text.

ROUGE-1 metric is employed, which compares the precision, recall, and F1-score of unigrams (individual words or tokens) in generated equations to reference equations. This metric allows us to assess the quality of the generated equations in terms of word-level overlap, revealing information about the system's capacity to repeat essential phrases and concepts.

b. ROUGE-2

Another evaluation metric often used in natural language processing and text summarizing tasks is the ROUGE-2. ROUGE-2, as compared to ROUGE-1, concentrates on bigrams, which are pairs of successive words or tokens. This metric compares the precision, recall, and F1-score of bigrams in generated text to those in reference text.

The formula for ROUGE-2 is as follows:

$$\text{ROUGE} - 2 = \frac{\text{Number of overlapping bigrams in generated and reference text}}{\text{Total number of bigrams in reference text}} \quad 5.4$$

- **Number of overlapping bigrams in generated and reference text:** This is the number of unique pairs of consecutive words that appear both in the generated text and the reference text.

- **Total number of bigrams in reference text:** This is the total number of unique pairs of consecutive words in the reference text.

c. ROUGE-L

ROUGE-L, or Longest Common Subsequence (LCS) based ROUGE, is a metric that measures the longest common subsequence of words to assess the similarity of generated text to reference text. ROUGE-L provides a more flexible evaluation than ROUGE-1 and ROUGE-2 by evaluating sequences of words that may not be contiguous but appear in the same order.

$$\text{ROUGE} - \text{L} = \frac{\text{Length of the longest common subsequence in generated and reference text}}{\text{Total number of words in reference text}} \quad 5.5$$

Furthermore, by focusing on the longest common subsequence (LCS) of words between generated and reference equations, the ROUGE-L metric provides insight into how effectively the generated equations replicate the structure and flow of the reference equations. It assesses the system's ability to reproduce meaningful word sequences, which is essential for mathematical problem solving.

iii. Final Answer Comparison

Finally, a final answer comparison is performed to determine the practical validity of the system's output. The produced equations are solved to obtain results, which are then compared to the expected or goal responses. This phase ensures that the system not only correctly formulates equations but also directs users to the correct solutions. The final response accuracy is a critical measure of the system's utility in real-world problem-solving contexts.

Finally, the evaluation system employs a multifaceted strategy that combines EM, ROUGE-1, ROUGE-2, ROUGE-L, and final response comparison. This complete

evaluation allows us to analyze the performance of math word problem-solving system by considering both the formulation of equations and the accomplishment of accurate solutions.

5.7 Development of Frontend and Backend

The user-friendly UI was developed using React, a JavaScript library. Upon training the model on the dataset, it was uploaded to the Hugging Face Hub, facilitating easy access to the fine-tuned model. The backend was constructed using Flask, a Python mini framework. The model was accessed using the Hugging Face Hub. The frontend was connected with an API, allowing users to input math word problems, which are then transferred to the backend. The model generates equations corresponding to the questions, and the API returns the generated equations to the frontend UI.

6 RESULTS AND ANALYSIS

6.1 Dataset Analysis

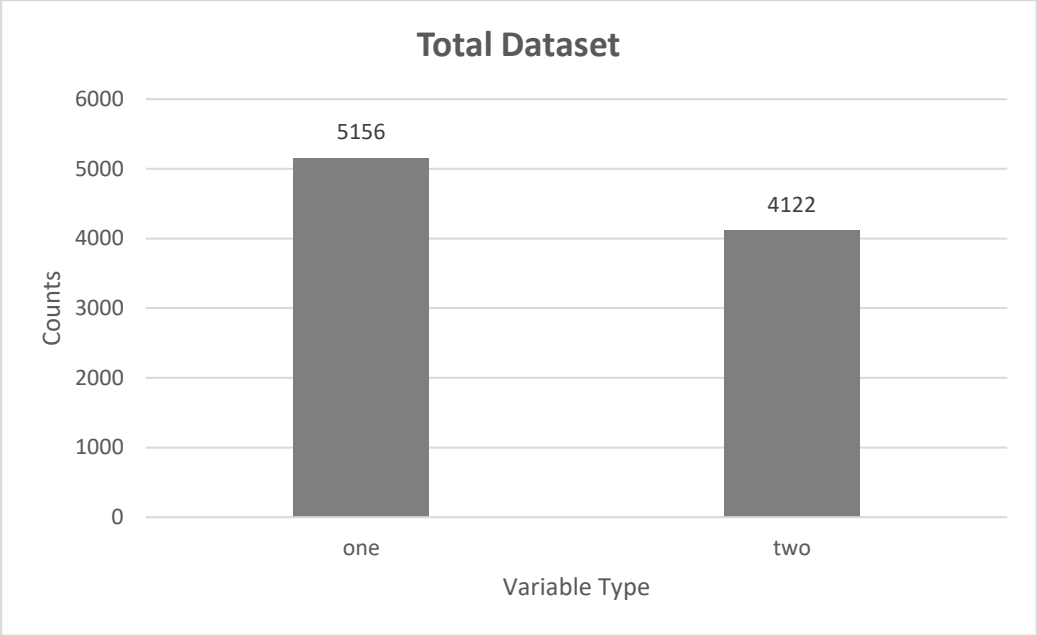


Figure 5- 1: Types of Variables in Dataset Initially

These 9,278 problems dataset was collected from various resources. Out of the total 9,278 data points in the dataset, the majority of the problems were categorized as one variable problems, accounting for 5156 instances. These one-variable problems involve mathematical questions that can be solved using a single variable, making them relatively simpler in nature. The dataset also contained 4122 instances of two variable problems, which are slightly more complex compared to one variable problems, requiring the use of two variables in their solutions. This distribution of one variable and two-variable provided valuable insights into the complexity and nature of mathematical problems present in the dataset, allowing for targeted model development and specialized algorithms tailored to each problem category.

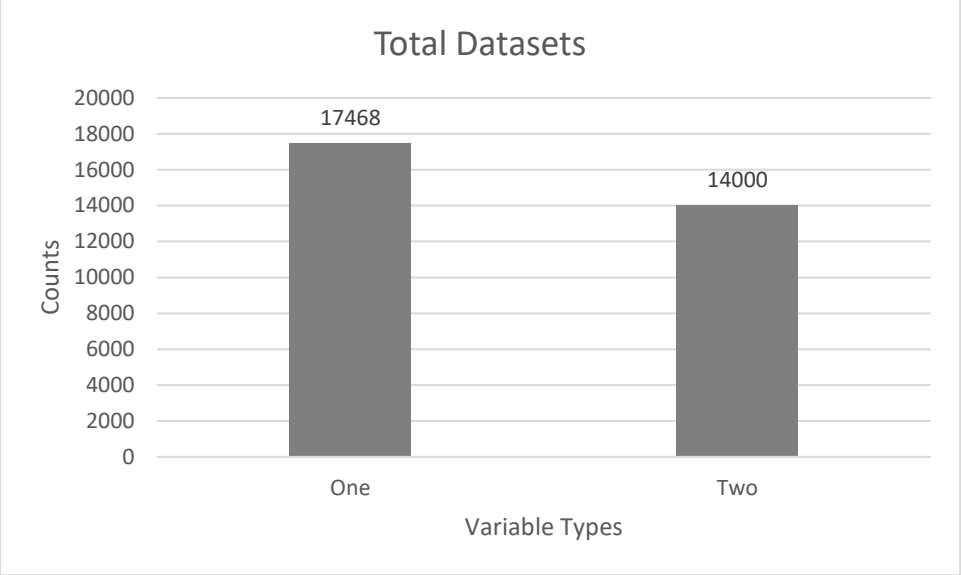


Figure 5- 2: Types of Variables in Dataset Final

After the addition of the dataset that was prepared, the total data points reached to 31,468 in the combined dataset. Out of which the single-variable problems are 17468 in number whereas two-variable problems were about 14000 in number following the total dataset.

6.2 Training Result Analysis

Trained T5 is trained on small dataset containing around 31,468 dataset using 5-fold cross validation and trained model on trained dataset and validate on the validation dataset after following the above-mentioned steps to train the model. The loss of first fold training and validation is shown below:

6.2.1 Training vs Validation loss

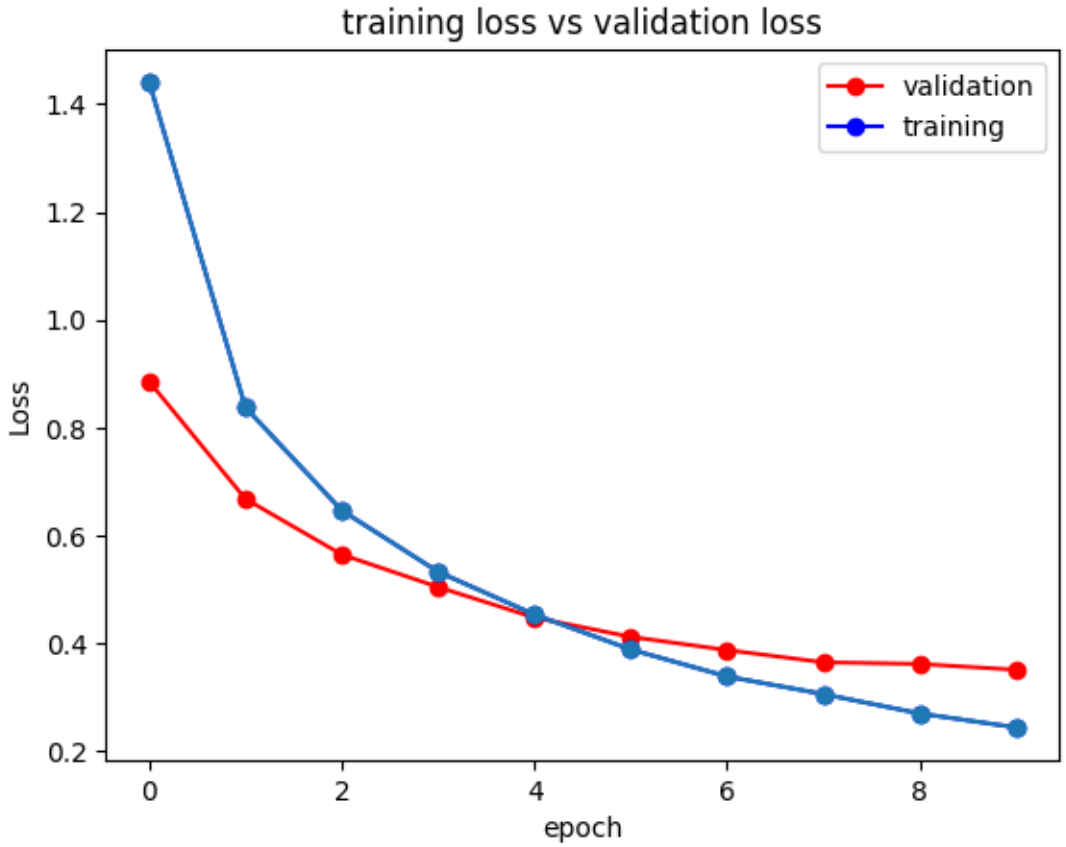


Figure 5- 3: Graph of Training vs Validation Loss in 10 epochs

The loss is calculated using the cross-entropy loss of model loss function. at initial training loss is high, then due to backpropagation on that dataset, loss reduces gradually but validation dataset is similar to training, so at first loss becomes low but after certain time when model learns more on training dataset, validation loss will be more than training loss.

To test the model performance on different epochs, the model is trained for the 20 epochs. This is the following graph of Training vs Validation Loss in 20 epochs

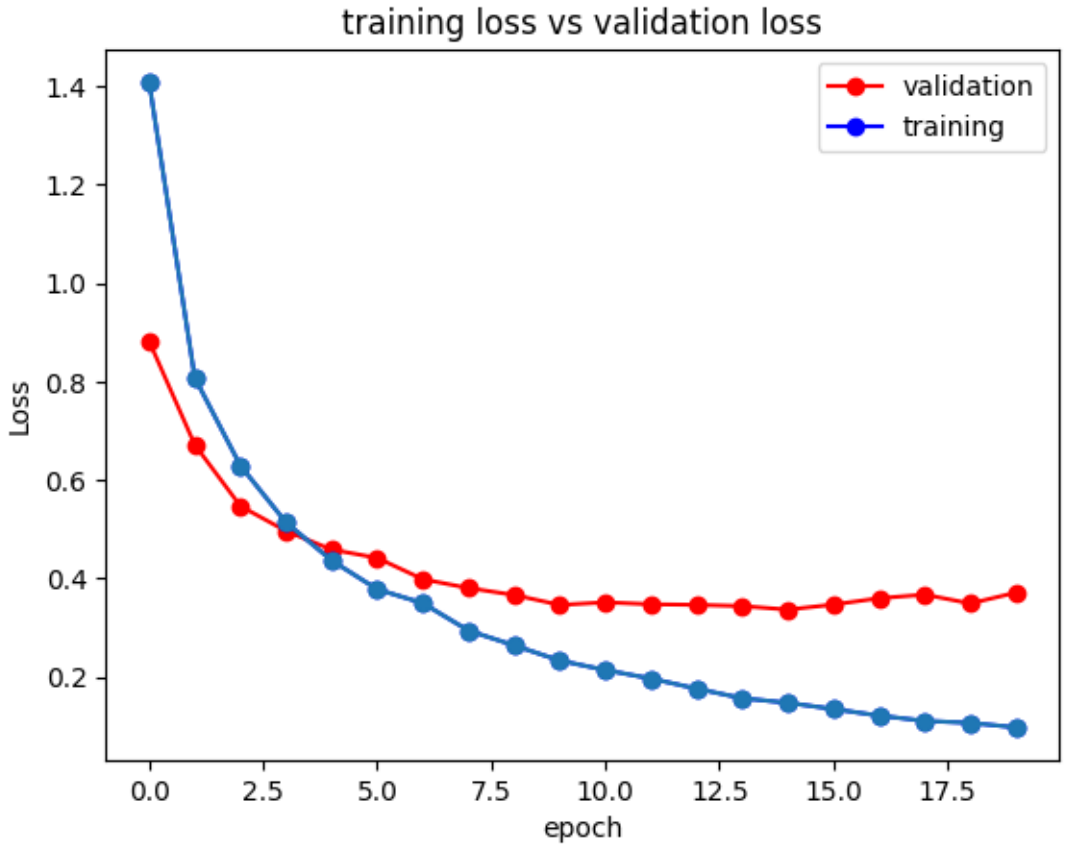


Figure 5- 4: Graph of Training vs Validation Loss in 20 epochs

From the above graph, training loss is observed to become too lower than validation loss while validation loss is kept increasing from the 10 epochs which suggest that model is overfitting on the seen data which is not good approach and it led to poor performance on the new data .so only 10 epochs was decided to be used while fine tuning the model.

The model is tested by providing some word problem. It is shown below:

Example 1

```
input_ids = tokenizer(
    "English to Math Equation:If you have five consecutive integers such that the even integers add up to 48 what is the smallest odd integer?",
    return_tensors="pt"
). input_ids.to(device)

outputs = model.generate(input_ids)
```

```
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

Output: x-2+x+x+2=48

Example 2

```
input_ids = tokenizer(
    "English to Math Equation:if 1/4 is added to the reciprocal of a number , the result is 5 more than 2 times the reciprocal of the original number. find the number",
    return_tensors="pt"
).input_ids.to(device)
outputs = model.generate(input_ids)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

Output: 1/4 + 1/n = 5 + 2(1/n)

Example 3

```
input_ids = tokenizer(
    "English to Math Equation:60 is 170% of what number?", return_tensors="pt"
).input_ids.to(device)
outputs = model.generate(input_ids)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

Output: 60 = 170/100*n

6.3 Accuracy Analysis

6.3.1 Accuracy Analysis for 9k Dataset

6.3.1.1 Exact Match

Table 5- 1: Exact Match Accuracy of 9k Dataset

Metrics	Accuracy (in %)
EM (Exact Match)	19.839

Exact Match accuracy is metric used to measure the percentage of correctly generated equations that match the expected or target equations exactly. An accuracy of 19.839% is achieved. It means that out of all generated equations, approximately 19.839% were a perfect match to the expected equations.

Equations can have several equivalent representations, and slight differences that do not impact the semantic meaning may cause an equation to be incorrect even though it is practically right. These variations are penalized by exact match metrics. So Exact match metrics is not suitable for the evaluation in context of equation generator.

6.3.1.2 ROUGE-1

Table 5- 2: ROUGE-1 Score of 9k Dataset

ROUGE-1	Score
Precision Score	0.924
Recall Score	0.878
F-measure Score	0.894

Precision Score (ROUGE-1): The precision score is calculated by comparing the fraction of correctly detected n-grams (individual words) in the generated text to the reference (anticipated) text. A score of 0.924 implies that about 92.4% of the n-grams in produced equations were found in the reference equations.

Recall score (ROUGE-1): It compares the fraction of correctly detected n-grams in the generated text to the reference text. A score of about 0.878 means that approximately 87.8% of the n-grams in the reference equations were properly detected in the produced equations.

ROUGE-1 F-Measure Score: The harmonic mean of precision and recall is the F-measure. It achieves a good combination of precision and recall. A score of around 0.894 indicates that generated equations achieve a decent combination of precision and recall.

6.3.1.3 ROUGE-2

Table 5- 3: ROUGE-2 Score of 9k Dataset

ROUGE-2	Score
Precision Score	0.79
Recall Score	0.746
F-measure Score	0.762

Precision Score (ROUGE-2): The ROUGE-2 precision score compares the fraction of correctly detected bigrams in the generated text to the reference text. A score of 0.790 indicates that about 79.0% of the bigrams in generated equations were located in the reference equations.

ROUGE-2 Recall Score: The ROUGE-2 recall score compares the proportion of correctly detected bigrams in the generated text to the reference text. A score of about 0.746 means that approximately 74.6% of the bigrams in the reference equations were correctly identified in generated equations.

ROUGE-2 F-Measure Score: The ROUGE-2 F-measure is the harmonic mean of precision and recall, offering a balance between the two. A score of around 0.762 indicates a good mix of precision and recall for bigrams in created equations.

6.3.1.4 ROUGE-L

Table 5- 4: ROUGE-L Score of 9k Dataset

ROUGE-L	Score
Precision Score	0.902
Recall Score	0.858
F-measure Score	0.873

Precision Score (ROUGE-L): The precision score is calculated by dividing the LCS of words between the generated text and the reference text by the length of the generated text. A score of 0.902 indicates that about 90.2% of the words in generated equations are part of the LCS with the reference equations.

Recall Score (ROUGE-L): The recall score is calculated by dividing the LCS of words between the generated text and the reference text by the length of the reference text. A score of roughly 0.858 implies that approximately 85.8% of the words in the reference equations are included in the LCS with generated equations.

ROUGE-L F-Measure Score: The ROUGE-L F-measure is the harmonic mean of precision and recall, providing a balanced measure of the LCS match. A score of about 0.873 indicates a good balance of precision and recall for the longest common subsequence of words between generated equations and the reference equations.

6.3.1.5 Answer Comparison

Table 5- 5: Answer Comparison Accuracy of 9k Dataset

Metrics	Accuracy(in %)
Final Answer Comparison	62.385

Measuring the correctness of generated equations is an important part the math word problem equation generator. However, this procedure presents certain difficulties, especially given that a single question or problem might frequently contain numerous

valid equations that lead to the correct answer. In such circumstances, depending exclusively on the equation as a measure for accuracy can lead to deceptive results.

An alternative approach is used to improve this problem and ensure a more thorough assessment of accuracy. Instead, then relying exclusively on the equations created by a system, the equations are solved to acquire their associated solutions. These answers are then compared to the expected or target replies, providing a more trustworthy measure of accuracy. This technique considers not only the formulation of the equations but also their practical utility in arriving at the correct solution by solving the equations and comparing the resulting solutions. It recognizes that different equations can result in the same result and assesses the system's ability to develop `equations that effectively direct towards the proper answer. The system's generated equations, when solved, led to correct answers in roughly 62.385% of the cases. It's a positive indicator of the system's effectiveness in addressing the task at hand.

6.3.2 Accuracy Analysis for 32k Dataset

After increasing our dataset from 9k to 32k, Massive change in the accuracy of the T5 can be observed.

6.3.2.1 Exact Match

Table 5- 6: Exact Match Accuracy of 32k Dataset

Metrics	Accuracy(in %)
EM(Exact Match)	52.76

An accuracy of 43.76% is achieved after the use of larger dataset. It means that out of all generated equations, approximately 43.76% were a perfect match to the expected equations. It can be concluded that increasing the dataset contributes in increasing accuracy of the T5 model.

6.3.2.2 Answer Comparison

Table 5- 7: Answer Comparison of 32k Dataset

Metrics	Accuracy (in %)
Final Answer Comparison	67.372

Similarly, in final answer comparison also, accuracy of T5 model tends to increase from 62.386% to 67.372%. So obviously, larger dataset contributes in better accuracy.

6.4 Accuracy Analysis of Flan-T5-large

The FLAN (Few-shot Language Adaptation Network) T5 model is a variant of the Text-To-Text Transfer Transformer (T5) architecture designed for few-shot learning tasks. It adapts T5 to effectively handle tasks where only a small amount of labeled data is available for training. FLAN T5 utilizes techniques such as meta-learning and data augmentation to improve performance on tasks with limited training examples.

Exact Match

Table 5- 8: Exact Match Accuracy of Flan-T5

Metrics	Accuracy(in %)
EM(Exact Match)	73.36

Final Answer Comparison

Table 5- 9: Answer Accuracy of Flan-T5

Metrics	Accuracy(in %)
EM(Exact Match)	76.38

Rouge-1 Comparison

Table 5- 10: ROUGE-1 Score of Flan- T5

ROUGE-1	Score by Flan-T5
Precision Score	0.957
Recall Score	0.9465
F-measure Score	0.9494

Rouge-2 Comparison

Table 5- 11: ROUGE-2 Score Flan-T5

ROUGE-2	Score
Precision Score	0.8759
Recall Score	0.8688
F-measure Score	0.8702

Rouge-l Comparison

Table 5- 12: ROUGE-L Score of Flan-T5

ROUGE-L	Score by Flan-T5
Precision Score	0.9371
Recall Score	0.9266
F-measure Score	0.9294

The improved performance of the FLAN T5 model compared to the standard T5 base model can be due to several factors. FLAN T5 is optimized for few-shot learning tasks, enabling it to utilize limited labeled data more effectively. Advanced data augmentation techniques are incorporated into FLAN T5, generating additional training examples from the limited data available, thus enhancing the model's ability to generalize. Meta-learning approaches are employed in FLAN T5, enabling the model to quickly adapt to

new tasks with minimal data. Furthermore, FLAN T5 may utilize regularization techniques to prevent overfitting on the limited training data, leading to improved generalization performance. Architectural modifications or hyperparameter adjustments specific to FLAN T5 contribute to its enhanced performance on few-shot learning tasks. Overall, the superior performance of FLAN T5 on few-shot learning tasks results from a combination of these factors, allowing it to make more effective use of limited labeled data compared to the standard T5 model.

6.5 Comparative Analysis Between T5 and BART

Bart is also Bidirectional and Auto-Regressive Transformers which is also sequence to sequence model like T5. Bart has some especial features like Denoising Autoencoder which means it's given a corrupted version of the original text and tasked with reconstructing it which allow Bart to learn underlying structure and relationship within the language which make it robust, Bidirectional encoder which captures contextual information from both sides. The parameter used in T5 base model is 220 million and parameter used in the Bart is 260 million.

Bart is tested with the math dataset to compare the performance between the models and to analysis the reason behind the performance variation. The base model is fine-tuned which is available at hugging face hub. On testing the model, following performance metric are found.

Exact Match

When we compared the equation generated by model and real equation using the comparison operator and got an accuracy of 49.05 while using Bart model which is much lesser than T5.

Table 5- 13: Exact Match Accuracy of Bart vs T5

Metrics	Accuracy by Bart (in %)	Accuracy by T5
EM (Exact Match)	49.05	43.76

Final Answer Comparison

Table 5- 14: Answer Accuracy of Bart vs T5

Metrics	Accuracy by Bart (in %)	Accuracy by T5
Final Answer Comparison	64.44	62.835%

Rouge-1 Comparison

Table 5- 15: ROUGE-1 Score of Bart vs T5

ROUGE-1	Score by Bart	Score by T5
Precision Score	0.955	0.924
Recall Score	0.88	0.878
F-measure Score	0.91	0.894

Rouge-2 Comparison

Table 5- 16: ROUGE-2 Score of Bart vs T5

ROUGE-2	Score by Bart	Score
Precision Score	0.81	0.790
Recall Score	0.75	0.746
F-measure Score	0.77	0.762

Rouge-l Comparison

Table 5- 17: ROUGE-L Score of Bart vs T5

ROUGE-L	Score by Bart	Score by T5
Precision Score	0.92	0.902
Recall Score	0.85	0.858
F-measure Score	0.87	0.893

The reason behind the variation of the accuracy score depends on the various factors. The main reason is slight variation of the architecture between two models, pre-training objectives. T5 is designed for wide range of NLP tasks and Bart is specifically pre-trained for denoising autoencoder tasks which is reason behind why T5 outperform the Bart Model. The higher accuracy of T5 might be due to its ability to capture more contextual information and generalize well across tasks.

6.6 Comparative Analysis using different RL approaches

RL was introduced to increase the accuracy of the system while generating equations. PPO was chosen as the RL algorithm. Two approaches were explored while applying reinforcement learning.

6.6.1 First Approach

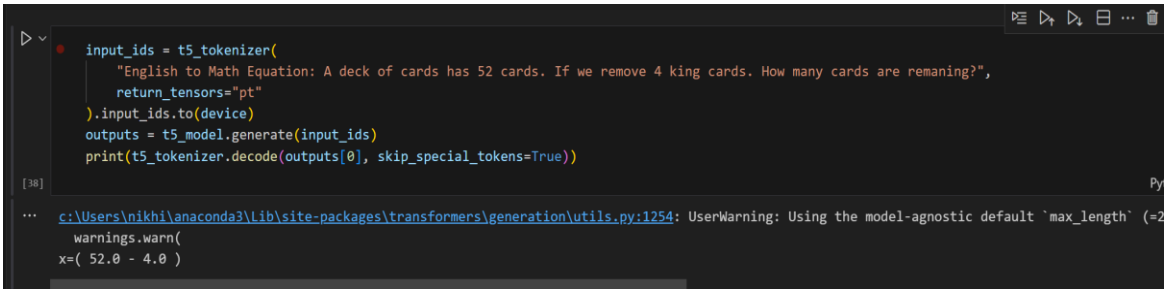
The first approach focused on creating a separate RL agent. In this approach, PPO agent was created which would learn to perform specific actions while given a state. This implementation consisted of custom environment were all the RL components were defined. The environment consisted of the current math word problem that the agent could interact with. The action space consisted of only one option generating equation to the given question using the fine-tuned T5 model. Observations space consisted of all the math word problem present in the dataset. The reward was assigned based on the similarity of the predicted equation with the true equations. Reward was assigned 1 if there was a match and 0 if there was no match. The agent was then trained and the output was not satisfactory. T5 model was used to encode and decode the given word problem based on the action the PPO agent chose. Since the action space consisted of only generating the entire equation the agent would essentially learn nothing and most of the action were performed by the T5 model.

This approach was then modified so that the action performed was not to generate the entire equation rather to predict each token. This increased the number of actions that the RL agent could perform when given a state. The number of actions were limited to the number of tokens that could be generated given a math word problem. The RL agent would choose among the given tokens to predict the equation to a given word problem. This approach was tested and the result were detrimental. The reward function applied

was similar to the first approach. The agent could not generate any tokens that are present in any given math equation. It would rather generate words/sentences that were completely unrelated. This approach was then dropped and new approach was tested to apply reinforcement learning.

6.6.2 Second Approach

New approach of applying reinforcement learning used T5 as an agent to generate math equations. Firstly, fine-tuned T5 was loaded and a PPO trainer was initialized. The reward function was similar to that of the first two approaches. The agent would get reward of 1 if the predicted equation matched the true equation and get a reward of 0 if the equation did not match. This approach was better than previous two equation as the model was able to generate equations to a given equation rather than generating random answers. The accuracy decreased than the fined tuned T5 model. This was due to the reward function checking for exact match and would always assign 0 if the equation did not match. Strict reward function caused the model to deviate from the generated token so the accuracy was affected severely.



6.6.3 Third Approach

For the third approach, the custom reward is generated based on the match of the generated equation with the true equation for a given question. This method followed the largest sequence matching that is, calculating the similarity ratio. Also, the model must generate length of the equation equal to the target equation, if it does not generate equal length then the model gets a penalty. For calculating the similarity ratio firstly, the longest common subsequence of characters that appear in the same order in both of the strings. Then the similarity ratio is calculated by taking the length of the longest common subsequence and dividing it by the length of the two sequences being compared. Final reward is calculated by multiplying the similarity ratio and the length

penalty. This reward function was a bit better than the one used before as the model as able to generate equations almost as the ground truth equation but this reward function also suffers from the problem of accuracy. The reward is between 0 and 1. Even if the model gets the reward of 0.9, the model answer is still wrong as our goal is to find the equation which must be completely correct, i.e. the model must get the reward of 1.

6.7 Comparative Analysis on Basis of Dataset

Table 5- 18: Comparative Analysis on Basis of Dataset

Model	**Exact Match Accuracy (in %)**	**Answer Comparison Accuracy (in %)**
T5-Base(9k dataset)	19.839	62.385
T5-Base(32k dataset)	**52.76**	**67.372**

The table shows a significant improvement in both exact match and response comparison accuracy, with gains of around 23% and 5%, respectively. This emphasizes the importance of dataset size in refining model correctness. The observed gains indicate that extending the dataset results in significant improvements in performance metrics. As a result, it can be concluded that a larger dataset considerably improves the model's capacity to precisely match answers and compare them to expected outcomes. This emphasizes the importance of dataset size in maximizing model performance and the advantages of using large datasets for training purposes.

6.8 Comparative Analysis on Basis of Model

Table 5- 19: Comparative Analysis on Basis of Model

Model	**Exact Match Accuracy (in %)**	**Answer Comparison Accuracy (in %)**
T5-Base	52.76	67.372
BART	49.05	61.44
Flan-T5-Large	**73.37**	**76.38**

The table shows the accuracy of different models in the same training dataset.

6.9 Result Analysis on Basis of RL

The results from t5-base and RL are:

Table 5- 20: Results from RL

reward_mean	0.855
reward_std	0.204
entropy	2.44

Table 5- 21: Accuracy Analysis of RL

Model	**Exact Match Accuracy (in %)**	**Answer Comparison Accuracy (in %)**
T5-Base (2k Dataset)	32	36.45
T5+RL	38.83	44.74

From Tables above, it is clearly understood that the RL has obviously improved the exact match accuracy and answer comparison accuracy of T5 by about 6% and 8% respectively.

6.10 Results from Different Models

```
question = '''At the Grocerry store apples are sold for $0.75 per pound.
Bananas are sold for $0.60 per pound. If Sarah wants to buy 3 pounds
of apples and 2 pounds of bananas, how much will she spend in total?'''
all_model_output(flan_t5_tokenizer,flan_t5_model,only_t5_tokenizer,
                 only_t5_model,RL_model,RL_tokenizer,bart_tokenizer,bart_model,question)
```

```
Flan t5 :  x = (3*0.75)+(2*0.6)

Base t5 5_epoch: x = (3+2)*0.75+0.60

RL t5 : x = (3+2)*0.75+0.60

Bart 10_epoch:  x=0.75*3+0.60*2
```

```
question = '''A carpenter buys eighteen planks of wood and fifteen bags of nails for nine-hundred.
Later, he buys 9 planks of wood and 12 bags of nails for $450.
Find the cost of each plank of wood and each bag of nails.'''
all_model_output(flan_t5_tokenizer,flan_t5_model,only_t5_tokenizer,
                 only_t5_model,RL_model,RL_tokenizer,bart_tokenizer,bart_model,question)
```

```
Flan t5 :  18x+15y=900, 9x+12y=450

Base t5 5_epoch: 18x+15y=900, 9x+12y=450

RL t5 : x=(18+12)/(12-1)

Bart 10_epoch:  x+y=18, 9x+12y=450
```

```
question = '''The perimeter of rectangle is 56 cm and area is 70. Determint the length of each side.'''
all_model_output(flan_t5_tokenizer,flan_t5_model,only_t5_tokenizer,
                 only_t5_model,RL_model,RL_tokenizer,bart_tokenizer,bart_model,question)
```

```
Flan t5 :  2x+2y=56, xy=70

Base t5 5_epoch: x=56*70

RL t5 : x=56+70

Bart 10_epoch:  x=56/2
```

```
[15] question = '''A movie ticket costs $12. If you have $48, how many tickets can you buy?'''
     all_model_output(flan_t5_tokenizer,flan_t5_model,only_t5_tokenizer,
                      only_t5_model,RL_model,RL_tokenizer,bart_tokenizer,bart_model,question)
```

```
Flan t5 :  x = 48/12

Base t5 5_epoch: x = 48-12

RL t5 : x = 48-12

Bart 10_epoch:  x = 12, y = 48
```

```
question = '''Aria is twenty-seven years older than Mason.
Twenty-five years ago, Mason's age was 21 years more than
twice the age of Aria. Calculate their current ages.'''
all_model_output(flan_t5_tokenizer,flan_t5_model,only_t5_tokenizer,
                 only_t5_model,RL_model,RL_tokenizer,bart_tokenizer,bart_model,question)
```

```
Flan t5 :  x=y+27, (y-25)=2(x-25)+21

Base t5 5_epoch: x=27+21

RL t5 : x = y + 27, x-25=2(y-25)+21

Bart 10_epoch:  x=y+27, y-25=2x-25)+21
```

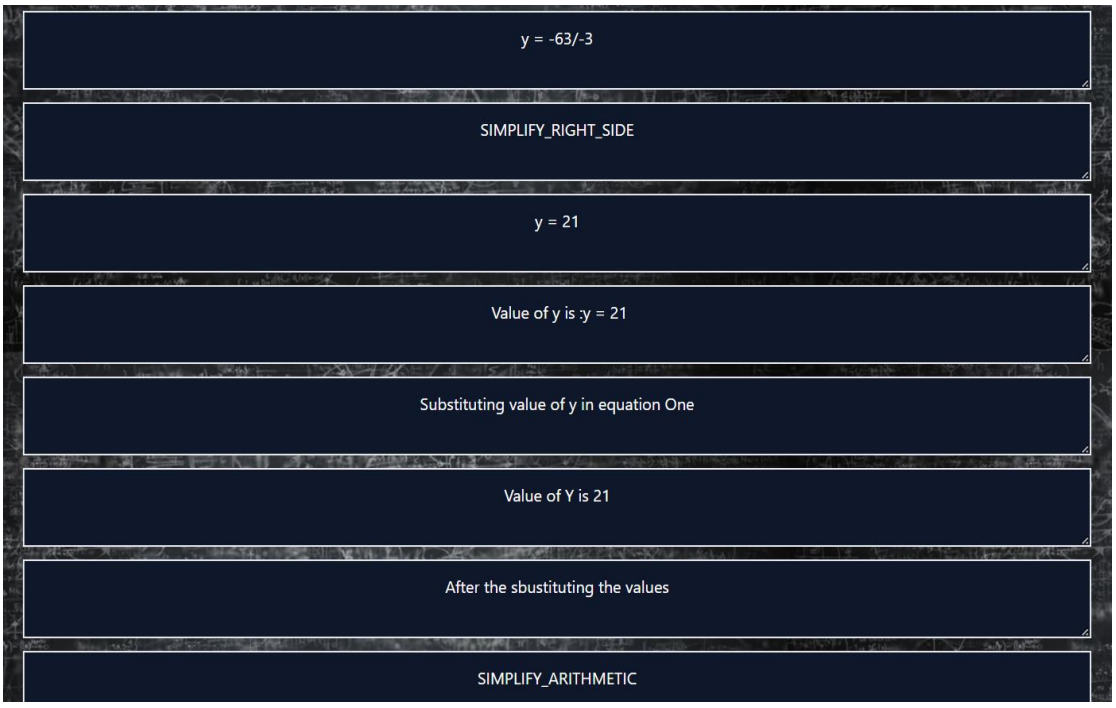
```
[17] question = '''A mother is three times as old as her daughter.
     Ten years ago, the model was four times as old as her daughter.
     Determine their current ages.'''
     all_model_output(flan_t5_tokenizer,flan_t5_model,only_t5_tokenizer,
                      only_t5_model,RL_model,RL_tokenizer,bart_tokenizer,bart_model,question)
```

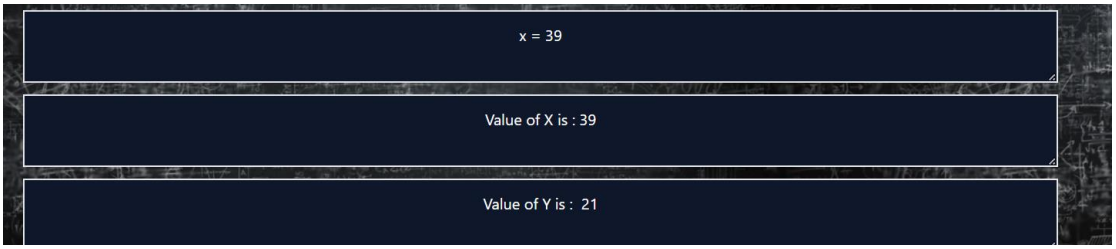
9s

```
Flan t5 :  x=3y, x-10=4(y-10)

Base t5 5_epoch: x=3y, x-10=4(y-10)

RL t5 : x=3y, x-10=4(y-10)

Bart 10_epoch:  x=3y, x-10=4(y-10)
```

6.11 UI Output Analysis


(-3y + 60) - 60 = -3 - 60
SIMPLIFY_LEFT_SIDE
-3y = -3 - 60
SIMPLIFY_ARITHMETIC
-3y = -63
DIVIDE_FROM_BOTH_SIDES
(-3y) / -3 = -63/-3
SIMPLIFY_LEFT_SIDE


y = -63/-3
SIMPLIFY_RIGHT_SIDE
y = 21
Value of y is :y = 21
Substituting value of y in equation One
Value of Y is 21
After the sbustituting the values
SIMPLIFY_ARITHMETIC


x = 39
Value of X is : 39
Value of Y is : 21

The UI Consists of an input field where the users can input their math word problem. The solution is generated when calculate button is pressed. After a few seconds, the equation for the corresponding question is generated. The solution of the question is generated step by step. Firstly, the solver finds each equation in terms of x i.e. if an equation is x + y = 60, it is converted to x = -y + 60. Then the equation is solved step by step.

6.12 Limitations

The objective of this project is to create a system capable of solving linear math word problems up to two variables. The system is intended to handle a wide range of inquiries from different domains. However, it is difficult to train the model comprehensively on all possible scenarios within these disciplines. The model can solve age-related difficulties, basic geometry questions, and simple speed-distance problems. However, it may struggle with unseen topics due to a lack of understanding of the individual circumstances within distinct fields. This limitation arises from the massive amount of data required to train the model on each potential case across several areas. The system can handle standard arithmetic word problems involving linear equations and up to two variables. However, due to the enormous dataset necessary to cover all possible scenarios, its performance may vary when questions are presented from different domains.

The model might not give the correct answer for new questions that are different from the ones it was trained on, like the given question with specific measurements not seen before, which could result in wrong answers.

Unseen question: Incorrect equation

```
# Unseen Geometry Question
question = '''A cylindrical container needs to hold exactly 500 cubic
             centimeters of water. Find the dimensions of the container
             if its height is 10 centimeters more than its radius.''';
Equation = math_solver(question)
```

```
x = 500/(3.14*x + 10)
```

Seen question: Correct equation

```
# Basic Geometry
question = '''The perimeter of a rectangle is 40 meters.
            If the length is 4 meters longer than the width,
            find the dimensions of the rectangle.''';
Equation = math_solver(question)
```

```
2x + 2y = 40, x = y + 4
```

Inability to handle math word problems involving more than two variables. For instance, the provided question requires consideration of three variables (Emily's, Mia's, and James's ages), exceeding the project's current support for only up to two variables.

```
# 3-variable question
question = '''Emily is currently twice as old as her sister Mia. Six years ago,
             Emily was three times as old as Mia. If their brother James is 5 years
             older than Emily, determine the current ages of Emily, Mia, and James.''';
Equation = math_solver(question)
```

```
x = 2y, x - 6 = 3(y - 6)
```

Inability to handle multi-step word problems with conditions requiring additional contextual understanding. For example, in below question, while the generated equation may be correct, the system provides answers for the present ages of the father and son instead of their ages after 3 years.

```
# Limitation
question = """The sum of present ages of father and son is 60. Six year's ago,
            the father's age was three times the age of son. what will be the
            age of son after 3 years.""";
Equation = math_solver(question)
```

```
x+y=60, x-6=3(y-6)
```

6.13 Time complexity of the T5 model

The time complexity of the transformer model depends on several factors like size of the inputs, number of layers and number of attention heads. To determine the complexity of the entire model, we need to compute the complexity for different parts of the model.

For Self-Attention Layer, the time complexity for the attention layer is O (n^2×d). Where n is the sequence length of the inputs and d is the dimension of the model. To calculate the self-attention value, all words in sequence needs to be compared with each other. For this comparison, a dot product needs to be computed which is proportional to d. For Feed-Forward Layer, the time complexity for the feed-forward layer is O (n×d^2). Where, n is the sequence length and d are the model dimension. The time complexity. For other operations, such as normalization and residual connections, have the lower computational complexity compared to the self-attention mechanism

So, Total time complexity=O (n^2 × d) + O (d^2 × n) =O (n^2)

7 FUTURE ENHANCEMENTS

The work presented in project can be extended to some extend like the domain expansion of math problems including varieties of problems in dataset. The generator's scope can be broadened to encompass other mathematical disciplines, including concepts from probability and statistics, ratio, profit loss and variety other topics. This upgrade makes it easier to generate a wide variety of mathematical questions, resulting in more engaging learning experiences for users. By including these extra areas, the generator can provide learners with thorough exposure to numerous mathematical topics, allowing them to develop versatile problem-solving skills and a deeper comprehension of mathematical principles.

The project can be further enhanced by launching it in the form of a mobile application software. This would enable faster access to the solution, and we could provide detailed solutions for the problems in chat format. Additionally, expanding the domain to cover problems with multiple variables would enhance the system's capabilities, as it currently only addresses problems with one or two variables.

8 CONCLUSION

Thus, a system was developed capable of solving algebraic linear math word problems involving one and two variables by translating them into precise mathematical equations with their solutions. Using diverse datasets and the power of transformer model the system was completed.

The dataset played a pivotal role in success of the project, as it determines the accuracy and relevance of the model. Despite facing the challenges such as limited English datasets and the Chinese format of Math23k, we persevered and collected a total 32k datasets by web scraping from various sources. These datasets covered specific problem categories such as age, arithmetic, numerical problems. Due to the limitation of dataset, the problem to tackle varieties of problem was limited. Number in the form of words were also accounted for so that the model can understand both the numeric as well as the alphabetic form of a number. Exploration of Reinforcement Learning was done, and a custom reward function was created to optimize its performance and evaluate the performance of the model.

One of the key features of our system is its ability to generate equations and provide step-by-step solutions. This not only assists users in understanding the given word problems but also helps them learn the techniques of equation solving. Students can utilize our system to compare their solutions and enhance their learning experience, while teachers can leverage it for teaching purposes, thereby augmenting the learning capacity of students.

In essence, our system excels in solving natural language word problems by accurately generating equations and presenting step-by-step solutions. This comprehensive approach distinguishes our system from others, providing users with a unique and effective tool for tackling mathematical challenges with ease and clarity.

APPENDICES

A. Project Timeline

Table 7- 1: Gantt chart with Project Activities and Timeline

ID	*Task Name*	*Start*	*Finish*	*Duration*	*2023*							*2024*	
					Jun	*Jul*	*Aug*	*Sep*	*Oct*	*Nov*	*Dec*	*Jan*	*Feb*
1	Project Planning	5/26/2023	6/19/2023	3w 4d									
2	Dataset Collection	5/26/2023	10/23/2023	21w 4d									
3	Data Preprocessing	7/23/2023	11/3/2023	14w 6d									
4	T5 Model Training	9/3/2023	11/20/2023	11w 2d									
5	RL Implementation	10/26/2023	1/10/2024	11w									
6	Model Evaluation	10/15/2023	1/23/2024	14w 3d									
7	Web Development	1/17/2024	2/16/2024	4w 3d									
8	Testing	10/15/2023	2/27/2024	19w 3d									
9	Documentation	5/26/2023	2/28/2024	39w 6d									

B. Headers and commands

In order to train the model, some of the headers and function that have been used are as follows

Pandas library: Pandas is an open-source python used for data manipulation and analysis. It provides easy to use data structure and data analysis tools for handling tabular data. Pandas can read and write data in many formats including csv, json.

transformer library: It is developed by Hugging Face which is open-source natural language processing library.

Pd.read_csv(): It is a function in the pandas that is used to read data from the csv and create a DataFrame object.

T5Tokenizer(): It is the class provided by the transformer library to instantiation a tokenizer object.

T5ForConditionalGeneration():It is the class provided by transformer to instantiate a pre-trained model.

From_pretrained(): It is the method provided by the T5Tokenizerclass.it allows to load a pre-trained tokenizer or model using a specific name.

Matplotlib.pyplot: It is python library used for plotting the graph and used for creating visualizations such as line plot, scatter plot etc.

SentencePiece: It is library which is used to tokenize text used in NLP generation task and based on BEP algorithm and T5Tokenizer is based on this library.

SymbolicPython: It is widely used for solving mathematical equations symbolically, making it a valuable tool for algebraic problem-solving in Python.

trl: trl is hugging library that provides set of tools to train transformer language models with reinforcement learning.

C. Details of Dataset

	A	B	C	D
	Question	Equation	Answer	Dataset
	Adam has $5.00 to buy an airplane that costs $4.2	x = 5.00-4.28	x = 0	one var
	2 beavers were working on their home. 1 went for	x = 2-1	x = 1	one var
	3 raccoons are playing in the woods. 2 go home to	x = 3-2	x = 1	one var
	To get to school, Chad must either walk around a c	x = 3.14-2	x = 1	one var
	Zoe picked five apples from her tree. Now the tree	x = 6-5	x = 1	one var
	A baker made 2 batches of chocolate chip cookies.	x = 2*3+4	x = 10	one var
	A museum had fourteen paintings. After they got r	x = 14-4	x = 10	one var
	A number decreased by 3 equals 7. What is the nu	x - 3 = 7	x = 10	one var
	A number multiplied by 10 equals 100. What is the	10x = 100	x = 10	one var
	A pack of chart paper costs $7.25. How much do 1	7.25x = 72.5	x = 10	one var
	A pack of index cards costs $2.25. How much do 1	2.25x = 22.5	x = 10	one var
	A pet store had 2 dogs. On Sunday they got 5 more	x = 2+5+3	x = 10	one var
	Adam put eighteen items in the shopping cart. Aft	x = 18-8	x = 10	one var
	Amy bought 7 pencils at the school store, but she	x = 7+3	x = 10	one var
	An architect was building his two story house. On	x = 6+4	x = 10	one var
	Betty also bought 140 shiny blue round stones. If 1	x = 140/14	x = 10	one var
	Dave was picking up sticks from his yard. He picke	x = 14-4	x = 10	one var

Figure 8 -1: Snapshot of Dataset of one variable

	A	B	C	D
1	Question	Equation	Answer	Dataset
2	Mike 's age , decreased by the age of his 4 year-old sister , is 11 . Mike 's age , increased b	x=4.0+11.0 , x+y	x=38,y= 15	Two var
3	The sum of the ages of a brother and sister is 27 . If four times the brother 's age is subtr	3.0*x-4.0*y=11.(	x=17,y= 10	Two var
4	the sum of a two-digit number is 9 . the number with the digits interchanged is 6 times t	10*x+y-6.0*x-6.(	x=4,y= 5	Two var
5	Mike invested $ 6000 for one year . He invested part of it at 9 % and the rest at 11 % . At	0.01*11.0*x+0.(	x=1800,y= 4200	Two var
6	One pan pizza and two cheesburgers provide 2860 calories . Two pan pizzas and one che	1.0*x+2.0*y=28(	x=1040,y= 910	Two var
7	Hollis is paying off two student loans . One loan charges 7 % interest per year . The other	0.01*7.0*x+0.01	x=4700,y= 3200	Two var
8	A car radiator has a 6-liter capacity . If the liquid in the radiators 40 % antifreeze , how m	40.0*x+100.0*y=	x=1,y= 5	Two var
9	How much 1 % boric acid solution and 5 % boric acid solution are needed to make 30 mL	1.0*x+5.0*y=30.	x=15,y= 15	Two var
10	The sum of two numbers is 2 . If the larger number is 14 more than three times the smal	x-3.0*y=14.0 , x+	x=5,y= -3	Two var
11	the sum of two numbers is 13 . one number is 1 more than twice the other . find the sma	x-2.0*y=1.0 , x+y	x=9,y= 4	Two var
12	the sum of two numbers is 81 the differance of the same two numbers is 9 . What are th	x+y=81.0 , x-y=9.	x=36,y= 45	Two var
13	The sum of two number is 17 . The second number is three times as much as the first nu	x+y=17.0 , y-3.0*	x=4.25,y= 12.75	Two var
14	Two numbers have a sum of 47 their difference is 5 . what are the two numbers ?	x+y=47.0 , x-y=5.	x=21,y= 26	Two var
15	If the price of copper is 65 cents/lb and the price of zinc is 30 cents/lb , how many pounc	65.0*x+30.0*y=7	x=30,y= 40	Two var
16	masc is 7 years older than sam and the sum their ages is 27 . What are their ages ?	x+y=27.0 , x-y=7.	x=10,y= 17	Two var
17	In a 2-digit number , the units digit is 4 more than the tens digit . If the digits are reverse	10*x+y-3.0*x-3.(	x=5,y= 1	Two var
18	Walt made an extra 9000 last year from a part time job . He invested part of the money a	0.01*9.0*x+0.01	x=4000,y= 5000	Two var
19	Mr. Wise bought $ 1950 worth of stock , some at $ 3.00 per share and some at $ 4.50 pe	3.0*x+4.5*y=19	x=400,y= 50	Two var
20	Christopher is 2 times as old as Gabriela . Nine years ago , Christopher was 5 times as old	x-5.0*y=-1*5.0*	x=24,y= 12	Two var

Figure 8 -2: Snapshot of the Dataset of two variable

D. Summary of Plagiarism Report

Linear Math Word Problem with T5 Model and Policy-based RL

ORIGINALITY REPORT

18%
SIMILARITY INDEX

PRIMARY SOURCES

1	www.researchgate.net Internet	365 words — 2%
2	elibrary.tucl.edu.np Internet	338 words — 2%
3	Pablo Ferri Borredà. "Deeep Continual Multimodal Multitask Modells for Out-of-Hospital Emergency Medical Call Incidents Triage Support in the Presence of Dataset Shifts", Universitat Politecnica de Valencia, 2024 Crossref Posted Content	158 words — 1%
4	www.coursehero.com Internet	131 words — 1%
5	Akshay Kulkarni, Adarsha Shivananda, Anoosh Kulkarni, Dilip Gudivada. "Applied Generative AI for Beginners", Springer Science and Business Media LLC, 2023 Crossref	128 words — 1%
6	github.com Internet	87 words — < 1%
7	arxiv.org Internet	82 words — < 1%

8	Seif Elmoushy, Mostafa Saeed, Wael H. Gomaa. "Empowering MBTI Personality Classification through Transformer-Based Summarization Model", 2023 Intelligent Methods, Systems, and Applications (IMSA), 2023 Crossref	80 words — < 1%
9	www.mdpi.com Internet	60 words — < 1%
10	"ECAI 2020", IOS Press, 2020 Crossref	59 words — < 1%
11	stackoverflow.com Internet	57 words — < 1%
12	Ekeh, Bryan. "The Guidance Signal Building Algorithm: A Method to Guide Neural Abstractive Summarization", University of Toronto (Canada), 2023 ProQuest	54 words — < 1%
13	ebin.pub Internet	52 words — < 1%
14	medium.com Internet	47 words — < 1%
15	Shokhikha Amalana Murdivien, Jumyung Um. "BoxStacker: Deep Reinforcement Learning for 3D Bin Packing Problem in Virtual Environment of Logistics Systems", Sensors, 2023 Crossref	43 words — < 1%
16	mdpi-res.com Internet	43 words — < 1%
17	www.freeprojectz.com Internet	42 words — < 1%

18	Sangam Aryal, Sangeeta Sharma, Siddhant Sedai, Prashraya Aryal, Jalauddin Mansur. "Brain Tumor Detection Using Convolutional Neural Networks (CNNs)", Kathford Journal of Engineering and Management, 2023 Crossref	41 words — < 1%
19	www.ics.uci.edu Internet	41 words — < 1%
20	www.mathaware.org Internet	40 words — < 1%
21	huggingface.co Internet	38 words — < 1%
22	www.ijitee.org Internet	38 words — < 1%
23	Abby Newcomb, Jugal Kalita. "Explaining Math Word Problem Solvers", Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval, 2022 Crossref	36 words — < 1%
24	m.moam.info Internet	35 words — < 1%
25	www.algebra.com Internet	35 words — < 1%
26	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2023 Crossref	34 words — < 1%
27	biblio.ugent.be Internet	31 words — < 1%

28	www.geeksforgeeks.org Internet	31 words — < 1%
29	"Proceedings of the 2nd International Conference on Data Engineering and Communication Technology", Springer Science and Business Media LLC, 2019 Crossref	29 words — < 1%
30	dokumen.pub Internet	29 words — < 1%
31	dspace.cuni.cz Internet	29 words — < 1%
32	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2021 Crossref	28 words — < 1%
33	par.nsf.gov Internet	28 words — < 1%
34	Mahdi Khodayar, Jacob Regan. "Deep Neural Networks in Power Systems: A Review", Energies, 2023 Crossref	27 words — < 1%
35	aclanthology.org Internet	27 words — < 1%
36	www.frontiersin.org Internet	27 words — < 1%
37	dl.lib.uom.lk Internet	25 words — < 1%
38	export.arxiv.org	

	Internet	25 words — < 1%
39	ir.nust.na Internet	25 words — < 1%
40	www.hindawi.com Internet	25 words — < 1%
41	www.politesi.polimi.it Internet	25 words — < 1%
42	Twumasi, Augustine. "Towards Reinforcement Learning Driven Mesh Adaptivity for Second Order Elliptic Problems", The University of Texas at El Paso ProQuest	24 words — < 1%
43	it.overleaf.com Internet	24 words — < 1%
44	www.merlin.uzh.ch Internet	24 words — < 1%
45	"Chinese Computational Linguistics", Springer Science and Business Media LLC, 2019 Crossref	23 words — < 1%
46	as-books.com Internet	23 words — < 1%
47	www.jlko.eu Internet	22 words — < 1%
48	riunet.upv.es Internet	21 words — < 1%

49	Abdelmoumin, Ghada A.. "Study of Distributed Low-Complexity Machine Learning-Enabled Intelligent IDS for Mission Critical IoT", Howard University, 2024 ProQuest	20 words — < 1%
50	Peerawat Chomphooyod, Atiwong Suchato, Nuengwong Tuaycharoen, Proadpran Punyabukkana. "English grammar multiple-choice question generation using Text-to-Text Transfer Transformer", Computers and Education: Artificial Intelligence, 2023 Crossref	20 words — < 1%
51	Chigozie Enyinna Nwankpa. "Advances in Optimisation Algorithms and Techniques for Deep Learning", Advances in Science, Technology and Engineering Systems Journal, 2020 Crossref	19 words — < 1%
52	Zajic, David M. "Multiple alternative sentence compressions as a tool for automatic summarization tasks", Proquest, 20111109 ProQuest	19 words — < 1%
53	aaltodoc.aalto.fi Internet	19 words — < 1%
54	repository.iaa.ac.tz:8080 Internet	19 words — < 1%
55	www.arxiv-vanity.com Internet	19 words — < 1%
56	Vanlalruata Hnamte, Jamal Hussain. "Enhancing security in software-defined networks: An approach to efficient ARP spoofing attacks detection and mitigation", Telematics and Informatics Reports, 2024 Crossref	18 words — < 1%

57	faculty.utrgv.edu Internet	18 words — < 1%
58	recerc.eu Internet	18 words — < 1%
59	upcommons.upc.edu Internet	18 words — < 1%
60	Carthon, Mark, III. "Dictionary-Based Data Generation for Fine-Tuning Bert for Adverbial Paraphrasing Tasks.", The University of Wisconsin - Milwaukee, 2020 ProQuest	17 words — < 1%
61	Xiaodong Xu, Huachao Xiong, Yining Wang, Yue Che, Shujun Han, Bizhu Wang, Ping Zhang. "Knowledge-enhanced semantic communication system with OFDM transmissions", Science China Information Sciences, 2023 Crossref	17 words — < 1%
62	Zhiwu Shang, Baoren Zhang, Wanxiang Li, Shiqi Qian, Jie Zhang. "Machine remaining life prediction based on multi-layer self-attention and temporal convolution network", Complex & Intelligent Systems, 2021 Crossref	17 words — < 1%
63	deepai.org Internet	17 words — < 1%
64	engineeringsarokar.com Internet	17 words — < 1%
65	sinergiejournal.eu Internet	17 words — < 1%

66	"Neural Information Processing", Springer Science and Business Media LLC, 2019 Crossref	16 words — < 1%
67	anrg.usc.edu Internet	16 words — < 1%
68	mafiadoc.com Internet	16 words — < 1%
69	Tianfan Jin, Qiyuan Zhao, Andrew B Schofield, Brett Matthew Savoie. "Machine Learning Models Capable of Chemical Deduction for Identifying Reaction Products", American Chemical Society (ACS), 2023 Crossref Posted Content	15 words — < 1%
70	flipkarma.com Internet	15 words — < 1%
71	Aigbavboa, Solomon Ohi. "Pharmaceutical Supply Chains in Nigeria: A Framework for Outsourcing Outbound Value Chains", University of Johannesburg (South Africa), 2021 ProQuest	14 words — < 1%
72	Hussain, Shehzeen Samarah. "Robust and Efficient Deep Learning for Multimedia Generation and Recognition", University of California, San Diego, 2023 ProQuest	13 words — < 1%
73	Lei Kang, Pau Riba, Marçal Rusiñol, Alicia Fornés, Mauricio Villegas. "Pay attention to what you read: Non-recurrent handwritten text-Line recognition", Pattern Recognition, 2022 Crossref	13 words — < 1%

74	fdocuments.net Internet	13 words — < 1%
75	www.semanticscholar.org Internet	13 words — < 1%
76	Hans Vonk. "Some trends in the development of curricula for the professional preparation of primary and secondary school teachers in Europe: A comparative study", British Journal of Educational Studies, 1991 Crossref	12 words — < 1%
77	Liqiang Jing, Xuemeng Song, Xuming Lin, Zhongzhou Zhao, Wei Zhou, Liqiang Nie. "Stylized Data-to-Text Generation: A Case Study in the E-Commerce Domain", ACM Transactions on Information Systems, 2023 Crossref	12 words — < 1%
78	Ruby Ansar, Prachi Upadhyay, Manish Singhal, Ashish Sharma, Manoj Singh Gaur. "Characterizing impacts of multi-Vt routers on power and reliability of Network-on-Chip", 2015 Eighth International Conference on Contemporary Computing (IC3), 2015 Crossref	12 words — < 1%
79	etheses.saurashtrauniversity.edu Internet	12 words — < 1%
80	ntnuopen.ntnu.no Internet	12 words — < 1%
81	repository.uin-suska.ac.id Internet	12 words — < 1%
82	"Advances in Artificial Intelligence", Springer Science and Business Media LLC, 2019 Crossref	11 words — < 1%

83	Jiao Shi, Tianyun Su, Xinfang Li, Fuwei Wang, Jingjing Cui, Zhendong Liu, Jie Wang. "A Machine-Learning Approach Based on Attention Mechanism for Significant Wave Height Forecasting", Journal of Marine Science and Engineering, 2023 Crossref	11 words — < 1%
84	Waseem Abbas, Zuping Zhang, Muhammad Asim, Junhong Chen, Sadique Ahmad. "AI-Driven Precision Clothing Classification: Revolutionizing Online Fashion Retailing with Hybrid Two-Objective Learning", Information, 2024 Crossref	11 words — < 1%
85	Zheni Zeng, Yi-Chen Nie, Ning Ding, Qian-Jun Ding, Wei-Ting Ye, Cheng Yang, Maosong Sun, Weinan E, Rong Zhu, Zhiyuan Liu. "Transcription between human-readable synthetic descriptions and machine-executable instructions: an application of the latest pre-training technology", Chemical Science, 2023 Crossref	11 words — < 1%
86	core.ac.uk Internet	11 words — < 1%
87	docslib.org Internet	11 words — < 1%
88	www.irjet.net Internet	11 words — < 1%
89	www.ncbi.nlm.nih.gov Internet	11 words — < 1%
90	Sukhavasi, Susrutha Babu. "Deep Neural Network Approach for Pose, Illumination, and Occlusion	10 words — < 1%

Invariant Driver Emotion Detection", University of Bridgeport, 2022
ProQuest

91	Vasudev Gohil, Satwik Patnaik, Hao Guo, Dileep Kalathil, Jeyavijayan Rajendran. "DETERRENT: Detecting Trojans Using Reinforcement Learning", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023 Crossref	10 words — < 1%
92	ar5iv.labs.arxiv.org Internet	10 words — < 1%
93	idoc.pub Internet	10 words — < 1%
94	pdffox.com Internet	10 words — < 1%
95	www.askpython.com Internet	10 words — < 1%
96	www.mlmi.eng.cam.ac.uk Internet	10 words — < 1%
97	Bilal Siyam, Amjad Abu Saa, Omar Alqaryouti, Khaled Shaalan. "Arabic Arithmetic Word Problems Solver", Procedia Computer Science, 2017 Crossref	9 words — < 1%
98	Jean Rabault, Miroslav Kuchta, Atle Jensen, Ulysse Réglade, Nicolas Cerardi. "Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control", Journal of Fluid Mechanics, 2019 Crossref	9 words — < 1%

99 Mizuho Nishio, Takaaki Matsunaga, Hidetoshi Matsuo, Munenobu Nogami et al. "Fully automatic summarization of radiology reports using natural language processing with large language models", Informatics in Medicine Unlocked, 2024 — 9 words — < 1%
Crossref

100 Ritvik Nair, Niyaz Ahmed, Nikhil Pavanan, Shailender Kumar. "Solving Arithmetic Word Problems Using Block Recurrent Transformer", 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2023 — 9 words — < 1%
Crossref

101 Suriyavarman S, Arockia Xavier Annie R. "Lung Nodule Segmentation and Classification using U-Net and Efficient-Net", International Journal of Advanced Computer Science and Applications, 2023 — 9 words — < 1%
Crossref

102 Ting Lu, Haitao Jiang, Shan Chang, Guohua Liu. "Improving Math Word Problems Solver with Logical Semantic Similarity", 2023 International Joint Conference on Neural Networks (IJCNN), 2023 — 9 words — < 1%
Crossref

103 Zhibin Yang, Linquan Xing, Zonghua Gu, Yingmin Xiao, Yong Zhou, Zhiqiu Huang, Lei Xue. "Model-Based Reinforcement Learning and Neural-Network-Based Policy Compression for Spacecraft Rendezvous on Resource-Constrained Embedded Systems", IEEE Transactions on Industrial Informatics, 2023 — 9 words — < 1%
Crossref

104 Zhijian Ou. "Energy-Based Models with Applications to Speech and Language Processing", Foundations and Trends® in Signal Processing, 2024 — 9 words — < 1%
Crossref

105	bora.uib.no Internet	9 words — < 1%
106	discovery.researcher.life Internet	9 words — < 1%
107	isprs-archives.copernicus.org Internet	9 words — < 1%
108	pt.scribd.com Internet	9 words — < 1%
109	staging-ai.jmir.org Internet	9 words — < 1%
110	su-plus.strathmore.edu Internet	9 words — < 1%
111	supportline.microfocus.com Internet	9 words — < 1%
112	www.researchsquare.com Internet	9 words — < 1%
113	"Business Process Management Forum", Springer Science and Business Media LLC, 2019 Crossref	8 words — < 1%
114	"Computational Intelligence in Communications and Business Analytics", Springer Science and Business Media LLC, 2022 Crossref	8 words — < 1%
115	Abdessamad Benlahbib, El Habib Nfaoui. "Aggregating customer review attributes for online reputation generation", IEEE Access, 2020 Crossref	8 words — < 1%

116 Abdullah, Deen Mohammad. "Query Focused Abstractive Summarization Using BERTSUM Model.", University of Lethbridge (Canada), 2020
ProQuest
8 words — < 1%

117 Dequan Zheng, Jia Hu, Yunxi Xie, Jingkai Li. "Math Word Problem Solving with Ensemble Learning", 2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), 2021
Crossref
8 words — < 1%

118 Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert et al. "ChatGPT for good? On opportunities and challenges of large language models for education", Learning and Individual Differences, 2023
Crossref
8 words — < 1%

119 Joyeeta Goswami, Kaushal Kumar Prajapati, Ashim Saha, Apu Kumar Saha. "Parameter-efficient fine-tuning large language model approach for hospital discharge paper summarization", Applied Soft Computing, 2024
Crossref
8 words — < 1%

120 Lukas Stankevičius, Mantas Lukoševičius, Jurgita Kapočiūtė-Dzikienė, Monika Briedienė, Tomas Krilavičius. "Correcting Diacritics and Typos with a ByT5 Transformer Model", Applied Sciences, 2022
Crossref
8 words — < 1%

121 Lulu Zhao, Weiran Xu, Chunyun Zhang, Jun Guo. "Leveraging speaker-aware structure and factual knowledge for faithful dialogue summarization", Knowledge-Based Systems, 2022
Crossref
8 words — < 1%

122 Madhusmita Sahu, Rasmita Dash. "Cognitive land cover mapping: A three-layer deep learning architecture for remote sensing data classification", Environmental Challenges, 2024 — 8 words — < 1%
Crossref

123 Minseok Kong, Jungmin So. "Empirical Analysis of Automated Stock Trading Using Deep Reinforcement Learning", Applied Sciences, 2023 — 8 words — < 1%
Crossref

124 Takehiro Murai, Yoshitaka Inoue, Assey Nambirige, George A. Annor. "Machine learning approach in predicting GlutoPeak test parameters from image data with AutoML and transfer learning", Heliyon, 2023 — 8 words — < 1%
Crossref

125 Wenbin Gan, Xinguo Yu, Mingshu Wang. "Automatic Understanding and Formalization of Plane Geometry Proving Problems in Natural Language: A Supervised Approach", International Journal on Artificial Intelligence Tools, 2019 — 8 words — < 1%
Crossref

126 Yuancheng Xia, Feng Li, Qing Liu, Li Jin, Zequn Zhang, Xian Sun, Lixu Shao. "ReasonFuse: Reason Path Driven and Global-Local Fusion Network for Numerical Table-Text Question Answering", Neurocomputing, 2022 — 8 words — < 1%
Crossref

127 Zhiwei Wang, Qi Lang, Xiaodong Liu, Wenlin Jing. "A multi-view graph learning model with dual strategies for solving math word problems", Neurocomputing, 2024 — 8 words — < 1%
Crossref

128 acikbilim.yok.gov.tr

	Internet	8 words — < 1%
129	epdf.pub Internet	8 words — < 1%
130	era.library.ualberta.ca Internet	8 words — < 1%
131	esnam.eu Internet	8 words — < 1%
132	fdocument.org Internet	8 words — < 1%
133	fdocuments.in Internet	8 words — < 1%
134	file.scirp.org Internet	8 words — < 1%
135	oa.upm.es Internet	8 words — < 1%
136	openbiomedicalengineeringjournal.com Internet	8 words — < 1%
137	repository.dl.itc.u-tokyo.ac.jp Internet	8 words — < 1%
138	repository.kulib.kyoto-u.ac.jp Internet	8 words — < 1%
139	uu.diva-portal.org Internet	8 words — < 1%
140	www.inettutor.com	

Internet

8 words — < 1%

141 www.slideshare.net
Internet

8 words — < 1%

142 Faranak Abri, Jianjun Zheng, Akbar Siami Namin, Keith S. Jones. "Markov Decision Process for Modeling Social Engineering Attacks and Finding Optimal Attack Strategies", IEEE Access, 2022
Crossref

7 words — < 1%

143 "Web Information Systems and Applications", Springer Science and Business Media LLC, 2021
Crossref

6 words — < 1%

144 Asma Al-Saleh, Mohamed Menai. "Solving Multi-Document Summarization as an Orienteering Problem", Algorithms, 2018
Crossref

6 words — < 1%

145 Jinghui Qin, Zhicheng Yang, Jiaqi Chen, Xiaodan Liang, Liang Lin. "Template-Based Contrastive Distillation Pretraining for Math Word Problem Solving", IEEE Transactions on Neural Networks and Learning Systems, 2024
Crossref

6 words — < 1%

146 Jordan Meadows, André Freitas. "Introduction to Mathematical Language Processing: Informal Proofs, Word Problems, and Supporting Tasks", Transactions of the Association for Computational Linguistics, 2023
Crossref

6 words — < 1%

147 Lecture Notes in Computer Science, 2005.
Crossref

6 words — < 1%

148 Murdock, Jaimie. "Topic Modeling the Reading and Writing Behavior of Information Foragers.",

6 words — < 1%

Indiana University
ProQuest

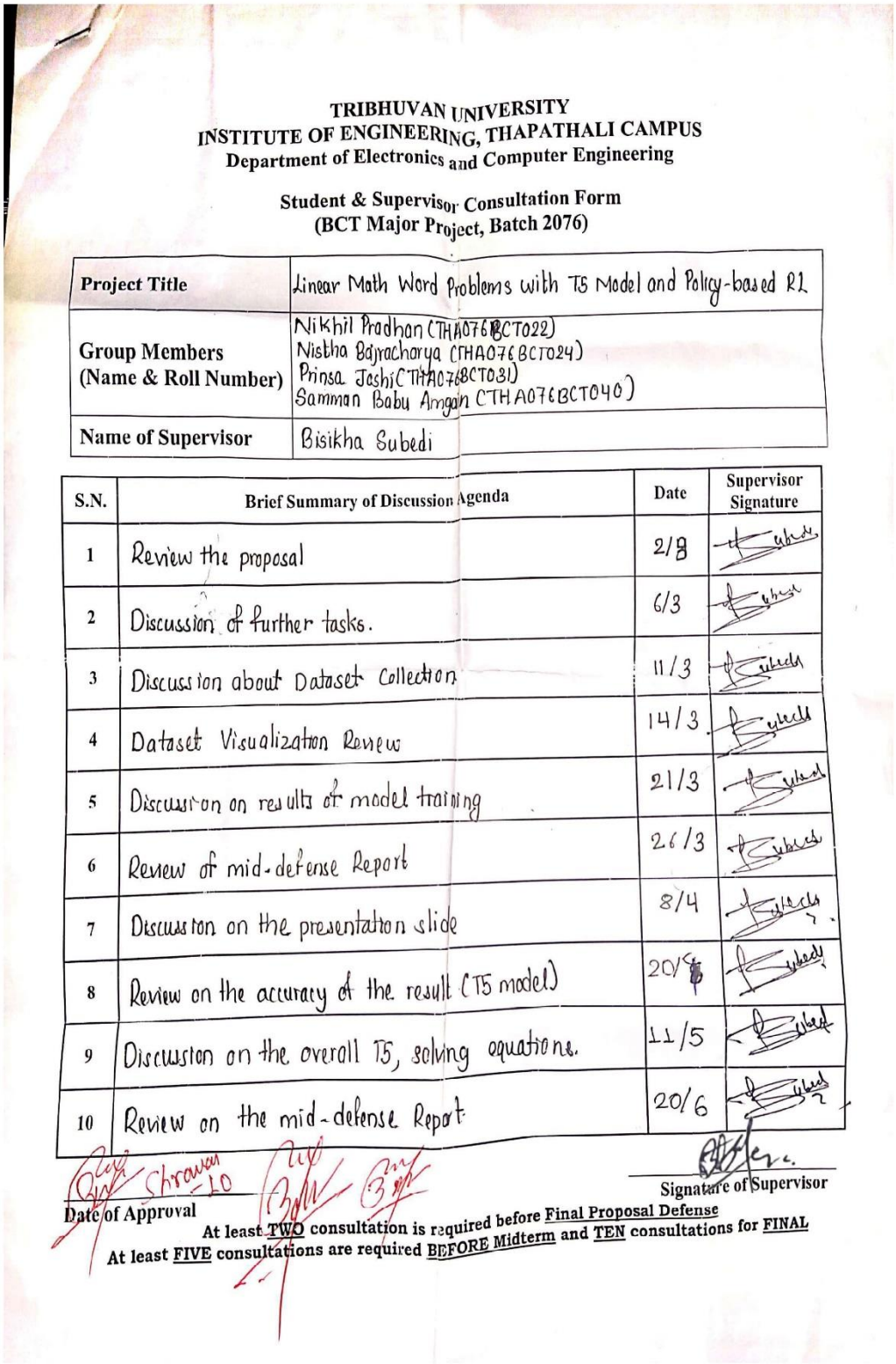
149 Yi Zhang, Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang. "Number-enhanced representation with hierarchical recursive tree decoding for math word problem solving", Information Processing & Management, 2024 — 6 words — < 1%
Crossref

150 Yuda Bi, Anees Abrol, Sihan Jia, Zening Fu, Vince D. Calhoun. "Gray Matters: An Efficient Vision Transformer GAN Framework for Predicting Functional Network Connectivity Biomarkers from Brain Structure", Cold Spring Harbor Laboratory, 2024 — 6 words — < 1%
Crossref Posted Content

151 Yun Ju, Jing Li, Guangyu Sun. "Ultra-Short-Term Photovoltaic Power Prediction Based on Self-Attention Mechanism and Multi-Task Learning", IEEE Access, 2020 — 6 words — < 1%
Crossref

EXCLUDE QUOTES ON
EXCLUDE BIBLIOGRAPHY ON
EXCLUDE SOURCES OFF
EXCLUDE MATCHES OFF

E. Student Supervisor Consultation Form



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING, THAPATHALI CAMPUS
Department of Electronics and Computer Engineering

Student & Supervisor Consultation Form
(BCT Major Project, Batch 2076)

Project Title	Linear Math Word Problems with T5 Model and Policy-based RL
Group Members (Name & Roll Number)	Nikhil Pradhan (THA076BCT022) Nistha Bajracharya (THA076BCT024) Prinsa Joshi (THA076BCT031) Samman Babu Amgain (THA076BCT040)
Name of Supervisor	Bisikha Subedi

S.N.	Brief Summary of Discussion Agenda	Date	Supervisor Signature
1	Review the proposal	2/8	
2	Discussion of further tasks.	6/3	
3	Discussion about Dataset Collection	11/3	
4	Dataset Visualization Review	14/3	
5	Discussion on results of model training	21/3	
6	Review of mid-defense Report	26/3	
7	Discussion on the presentation slide	8/4	
8	Review on the accuracy of the result (T5 model)	20/8	
9	Discussion on the overall T5, solving equations.	11/5	
10	Review on the mid-defense Report	20/6	

Shrawan -10

Date of Approval

Signature of Supervisor

At least TWO consultation is required before Final Proposal Defense
At least FIVE consultations are required BEFORE Midterm and TEN consultations for FINAL

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING, THAPATHALI CAMPUS
Department of Electronics and Computer Engineering
Student & Supervisor Consultation Form
(BCT Major Project, Batch 2076)

Project Title	Linear Equation (Up to Two variables) Word Problems with T5 Model and Policy-based RL
Group Members (Name & Roll Number)	Nikhil Pradhan (THA076 BCT022) Nistha Bajracharya (THA076 BCT024) Prinsa Joshi (THA076 BCT031) Samman Babu Amgain (THA076 BCT
Name of Supervisor	Er. Bibat Thokar

S.N.	Brief Summary of Discussion Agenda	Date	Supervisor Signature
1	Ongoing project discussion - RL and T5 Model	2080 -09-20	
2	Remarks for mid-report and slides	2080 -09-23	
3	Discussion on project accuracy and performance	2080 -10-21	
4	Remarks on the report, Use-case, entity, state, activity diag	2080 -10-25	
5	Final discussion on presentation and the report.	2080- 10-27	
6	Discussion on solver for one, two variable equation generator	2080-11 -04	
7	Discussion and feedback on the output, limitations of model	2080-11 -10	
8	Remarks on the report - conclusion, future enhancement, abstract	2080-11 -17	
9	Remarks for final slide presentation ppt.	2080-11 -20	
10	Final output evaluation and preparation for defense	2080-11 -22	

Jan-11

Date of Approval

Signature of Supervisor

At least TWO consultation is required before 1st Checkpoint Defense
At least FIVE consultations are required BEFORE 2nd Checkpoint and TEN consultations for FINAL

References

[1] B. Amnueypornsakul and S. Bhat, "Machine-guided Solution to Mathematical Word Problems," Department of Linguistics, Chulalongkorn University, 2014.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, \. Kaiser and I. Polosukhin, "Attention is All You Need," in *Curran Associates Inc.*, 2017.

[3] Y. Wang, X. Liu and S. Shi, "Deep Neural Solver for Math Word Problems," Association for Computational Linguistics, 2017.

[4] D. Huang, J. Liu, C.-Y. Lin and J. Yin, "Neural Math Word Problem Solver with Reinforcement Learning," Association for Computational Linguistics, 2018.

[5] L. Wang, D. Zhang, J. Zhang, X. Xu, L. Gao, B. T. Dai and H. T. Shen, "Template-Based Math Word Problem Solvers with Recursive Neural Networks," 2019.

[6] Rumshisky, Y. Meng and Anna, "Solving Math Word Problems with Double-Decoder Transformer," 2019.

[7] Q. Wu, Q. Zhang, Z. Wei and X. Huang, "Math Word Problem Solving with Explicit Numerical Values," Association for Computational Linguistics, 2021.

[8] Z. Jie, J. Li and W. Lu, "Learning to Reason Deductively: Math Word Problem Solving as Complex Relation Extraction," 2022.

[9] A. Wangperawong, "Attending to Mathematical Language with Transformers," *ArXiv*, Vols. @article{Wangperawong2018AttendingTM,, 2018.

[10] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," JMLR.org, 2020.

[11] D. Zhang, W. Li and S. Mumtaz, "An Improved Math Word Problem (MWP) Model Using Unified Pretrained Language Model (UniLM) for Pretraining," Hindawi Limited, 2022.

[12] B. Krishnamachari and M. Zong, "Solving Math Word Problems concerning Systems of Equations with GPT-3," 2023.

[13] J. Shen, Y. Yin, L. Li, L. Shang, X. Jiang, M. Zhang and Q. Liu, "Generate & Rank: A Multi-task Framework for Math Word Problems," in *Association for Computational Linguistics*, Punta Cana, Dominican Republic, 2021.