

**TRIBHUVAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**

**THAPATHALI CAMPUS**

**A**

**Major Project Proposal**

**On**

**Linear Equation (Up to Two Variables) Word Problems**

**with T5 Model and Policy-based RL**

**Submitted By:**

Nikhil Pradhan (THA076BCT022)

Nistha Bajracharya (THA076BCT024)

Prinsa Joshi (THA076BCT031)

Samman Babu Amgain (THA076BCT040)

**Submitted To:**

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

June, 2023

# 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

Nikhil Pradhan (THA076BCT022)

Nistha Bajracharya (THA076BCT024)

Prinsa Joshi (THA076BCT031)

Samman Babu Amgain (THA076BCT040)

ABSTRACT

This project aims to develop a system that solves mathematical word problems involving linear equations up to two variables using the T5 (Text-To-Text Transfer Transformer) model and a policy-based reinforcement learning (RL) approach. The system tackles the challenge of understanding and solving algebraic word problems by leveraging the power of natural language processing and machine learning techniques. The project's methodology involves fine-tuning the T5 model, a state-of-the-art language model, on a dataset of mathematical word problems and their corresponding solutions. The T5 model is trained to transform natural language problem statements into algebraic equations or equation systems, representing the underlying mathematical relationships. To enhance the solver's performance, a policy-based RL approach is incorporated. The RL component learns to take appropriate actions based on the problem statements, generating equations or selecting operations to solve the given problems. The RL agent receives rewards or penalties based on the correctness of its solutions, and through iterative training, it improves its problem-solving abilities. The expected output of the system includes not only the final answers to the algebraic word problems but also a detailed step-by-step solution process. This enables users to understand and verify the system's problem-solving logic, fostering transparency and trust. The effectiveness of the developed system will be evaluated through extensive testing on a diverse range of algebraic word problems. The system's performance will be measured based on accuracy, speed, and the ability to handle different problem structures and complexities.

*Keywords: Algebra, Linear Equations, Machine learning, Mathematical equations, NLP (Natural Language Processing), Policy-based reinforcement learning, Problem-solving, RL environment, T5 model, Variable identification*

# Table of Contents

[ACKNOWLEGEMENT i](#_Toc138059601)

[ABSTRACT ii](#_Toc138059602)

[List of Figures v](#_Toc138059604)

[List of Tables vi](#_Toc138059605)

[List of Abbreviations vii](#_Toc138059606)

[1. INTRODUCTION 1](#_Toc138059607)

[1.1 Background 1](#_Toc138059608)

[1.2 Motivation 2](#_Toc138059609)

[1.3 Problem Definition 3](#_Toc138059610)

[1.4 Objectives 3](#_Toc138059611)

[1.5 Scope and Application 4](#_Toc138059612)

[2. LITERATURE REVIEW 7](#_Toc138059613)

[3. SYSTEM ARCHITECTURE AND METHODOLOGY 11](#_Toc138059614)

[3.1 System Block Diagram 12](#_Toc138059615)

[3.2 Dataset Processing 13](#_Toc138059616)

[3.3 Transformer Architecture 14](#_Toc138059617)

[3.3.1 Input Layer 15](#_Toc138059618)

[3.3.2 Encoder Stack 15](#_Toc138059619)

[3.3.3 Self-Attention Mechanism 15](#_Toc138059620)

[3.3.4 Multi-Head Attention 17](#_Toc138059621)

[3.3.5 Feed-Forward Network 17](#_Toc138059622)

[3.3.6 Layer Normalization 17](#_Toc138059623)

[3.3.7 Decoder Stack 17](#_Toc138059624)

[3.3.8 Output Layer 17](#_Toc138059625)

[3.4 T5 Model Architecture 18](#_Toc138059626)

[3.5 T5 Model Training 19](#_Toc138059627)

[3.6 Optimization 20](#_Toc138059628)

[3.7 Implementation Tools 23](#_Toc138059629)

[3.7.1 Hardware Requirements 23](#_Toc138059630)

[3.7.2 Software Requirements 23](#_Toc138059631)

[4. EXPECTED OUTCOME 24](#_Toc138059632)

[4.1 Input problem into T5 model 24](#_Toc138059633)

[4.2 Output of t5 model 24](#_Toc138059634)

[4.3 Reinforcement Learning (RL) optimization 24](#_Toc138059635)

[4.4 RL-optimized equations 24](#_Toc138059636)

[4.5 Equation Solver Library 24](#_Toc138059637)

[4.6 Final steps and answers 25](#_Toc138059638)

[5. PROJECT SCHEDULE 26](#_Toc138059639)

[6. FEASIBILITY STUDY 27](#_Toc138059640)

[6.1 Operational Feasibility 27](#_Toc138059641)

[6.2 Economic Feasibility 27](#_Toc138059642)

[6.3 Technical Feasibility 27](#_Toc138059643)

[6.4 Legal Feasibility 27](#_Toc138059644)

[References 28](#_Toc138059645)

# List of Figures

[Figure 3 1: System Block Diagram of Linear Word Problem Solver 19](#_Toc138059606)

[Figure 3 2: Transformer Architecture Model 22](#_Toc138059607)

# List of Tables

[Table 5 1: Gantt chart with Project Activities and Timeline 33](#_Toc138059605)

# List of Abbreviations

|  |  |
| --- | --- |
| AI | Artificial Intelligence |
| RL | Reinforcement Learning |
| RNN | Recurrent Neural Network |
| LSTM | Long-Short Term Memory |
| GRU | Gated Recurrent Units |
| MLE | Maximum Likelihood Estimation |
| NLP | Natural Language Processing |
| T5 | Text-To-Text Transfer Transformer |
| CNN | Convolutional Neural Network |
| API | Application Programming Interface |
| GPU | Graphics Processing Unit |
| MSE | Mean Squared Error |
| LHS | Left-Hand Side |
| RHS | Right-Hand Side |
| BERT | Bidirectional Encoder Representations from Transformer |

|  |  |
| --- | --- |
|  |  |
|  |  |

# INTRODUCTION

## Background

Algebraic math word problems are an important aspect of math education and real-world problem solving. These challenges require humans to comprehend and transform verbal descriptions into mathematical equations by applying algebraic concepts to real-world situations. Solving algebraic word problems improves not only math skills but also critical thinking and problem-solving ability. Algebraic word problems can be used to a variety of situations, such as age problems, distance-rate-time questions, mixing problems, and so on. They usually contain variables, unknowns, and relationships between quantities, making them difficult to solve without a rigorous approach. Understanding the problem, discovering important information, building up equations or expressions, and solving for the unknowns are common procedures in solving algebraic word problems. However, many students and individuals struggle with these issues because they have difficulty understanding the terminology, detecting the underlying mathematical structure, or constructing the required equations.

To overcome these obstacles and give appropriate support, numerous tools and strategies for solving algebraic word problems have been developed. Traditional ways include problem-solving techniques such as making tables, drawing diagrams, or breaking the problem down into smaller components. Educators also use instructional strategies to teach problem-solving heuristics and provide opportunity for practice.

In recent years, technology and artificial intelligence (AI) have played an increasingly important role in assisting with algebraic math word problem solving. Researchers have investigated the use of machine learning and natural language processing techniques to create automated systems that can understand and solve algebraic word problems. These systems process and analyze textual descriptions of algebraic word problems using machine learning models such as deep neural networks and transformers. These models can learn to recognize patterns, extract significant information, and develop accurate mathematical representations by being trained on vast datasets of annotated issues. Furthermore, advances in reinforcement learning have made it possible to improve the performance of algebraic math word problem solvers. These systems can iteratively learn from feedback and enhance their problem-solving abilities over time by adding reinforcement learning techniques.

Furthermore, the incorporation of AI and machine learning approaches in algebraic math word problem solvers has the potential to resolve some of the most prevalent problems encountered by students and individuals. These systems can deliver individualized feedback, coaching, and step-by-step solutions to learners based on their specific needs and ability levels. They can also provide interactive interfaces that let users to enter and alter problem parameters, allowing them to gain a better grasp of the problem-solving process.

Automated algebraic math word problem solvers have the benefit of being accessible at all times and locations, offering prompt assistance and lowering reliance on human aid. They can be useful teaching tools that provide students the freedom to hone their individual problem-solving abilities. Additionally, the development of AI-driven algebraic math word problem solutions advances the field of instructional technology more broadly. It creates new opportunities for research into individualized instruction, intelligent tutoring systems, and adaptive learning systems. These systems can gather information on learners' performance, pinpoint their weak points, and provide tailored interventions, resulting in a more efficient and interesting learning process.

## Motivation

The ambition to improve the usability and effectiveness of education in mathematics serves as the driving force behind the creation of an Linear math word problem solver. It is well recognized that algebraic word problems present a considerable challenge for many students, frequently resulting in dissatisfaction and a lack of faith in their mathematical prowess. By developing an automatic solver, we hope to give students 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 algebraic concepts by providing step-by-step solutions, explanations, and tailored feedback.

Additionally, an 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. This individualized approach encourages student involvement, creates a good learning atmosphere, and equips students with the tools they need to overcome their struggles with algebraic word problems. In the end, the creation of an efficient and approachable algebraic math word problem solution can help to enhance mathematics education outcomes, boost student confidence, and foster a lifetime appreciation for the subject.

## Problem Definition

The issue we want to solve 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 to be able to examine the written descriptions, pick out important details, and produce precise mathematical equations or expressions. By developing such a solver, we hope to increase accessibility, effectiveness, and enjoyment for students and individuals who answer linear math word problems, enabling them to hone their problem-solving abilities and lay a solid basis in algebraic reasoning.

## Objectives

The main objectives of our project are listed below:

* To automate the process of solving linear word problems containing up to Two Variables.
* To achieve high-performance levels comparable to human experts in solving linear equation math word problems.

## Scope and Application

The scope of this project is to create a linear math word problem solver that uses transformer models, such as T5, to automate the process of solving algebraic word problems. The project intends to address a wide range of algebraic word problem types, such as age problems, distance-rate-time problems, mixture problems, and others. It focuses on utilizing the ability of transformer-based systems to interpret and process textual descriptions of algebraic word problems, extract significant information, and build accurate mathematical representations.

The target problem for this project is up to two variables algebraic, arithmetic, profit and loss problems.

* **Arithmetic:** Arithmetic problems involve the act of performing various operations like addition, multiplication, and subtraction on numbers. Some of the examples of arithmetic problems are:
  + The total age of two kids in a family is 27 years. What will be the total of their ages after three years?
  + The cost of one apple is Rs. 24, Find the cost of 15 such apples.
* **Algebraic:** Algebraic problems involve finding the unknown. Algebraic problems are solved by establishing necessary equations according to the problem and solving the equation to get the required value. Some example problems are:
  + When 2 is added to the product of 6 and a certain number, the result is 20. What is the value of the number?
  + The 22 students in Ms. Smith’s 2nd grade class each have a sibling or a pet. If 14 students have a sibling and 18 students have a pet, how many students have both a sibling and a pet?
  + Elenor bought 5 pints of frozen yogurt and a tray of jumbo shrimp from the Food Place for a total of $45. If the price of a tray of jumbo shrimp is $25, what is the price of a pint of frozen yogurt?
* **Profit and Loss:** Profit and Loss types of problem are related to problems that can be solved using a profit-loss formula to determine the price of a commodity in the market and understand how profitable a business. Some example problems are:
  + Suppose a shopkeeper has bought 1 kg of apples for 100 rs. And sold it for Rs. 120 per kg. How much is the profit gained by him?
  + A man buys a fan for Rs. 1000 and sells it at a loss of 15%. What is the selling price of the fan?
  + A dishonest dealer sells goods at a 10% loss on cost price but uses 20% less weight. Compute profit or loss percentage.

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

* **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.
* **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.
* **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.
* **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.

# 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. [1] The study conducted at 2014 by Bussaba Amnueypornsakul and Suma Bhat 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. 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. The equation was generated by creating three labels in the problem which were Change, Given and Result. The equation was in the form of (quantity in Given) + (quantity in Change) = Result. 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.

Template-Based Math Word Problem Solvers (Lei Wang,Dongxiang Zhang,2019) [2] was designed to solve arithmetic math word problems. 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 was tested upon Math23k which contained 23,64 problems and MAWPS which contained 2,373 problems. They achieved an accuracy of 66.9% in Math23k and 66.8% in MAWPS. The model could not perform well on problems that would generate long templates. Using modern NLP, Semantic understanding could also be improved.

Deep Neural Solver for Math Word Problems, Yang Wang [3] proposed to directly translate the math word problem into equation templates using an RNN instead of using previously used statistical learning approaches. 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. The RNN-based seq2seq model took the math word problem as the input and would generate an equation template (e.g., x = n1+n2+n3-n4) as an output. 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. Similarity between testing problem and problems available in the dataset is calculated by using Jaccard similarity. Equation template having the highest similarity score with the test problem is chosen. The model was tested on Math23k dataset and Alg514 and got an accuracy of 64.7% on Math23k and 70.1% on Alg514. Use of some other methods on small data sets could be used to increase accuracy further.

Neural Math Word Problem Solver [4]. A reinforcement learning approach to increase the accuracy of the result was proposed by Danqing Huang. 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 was then tested on 3 datasets Algebra514, Number Word Problem and Dolphin18k. The model outperformed all the other models following the hybrid approach and applying Reinforcement Learning.

A novel approach to solving math word problems was introduced by Qinzhou Wu [5] that focused on explicit numerical values. 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. 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.

A Deductive Reasoning Model for Math Word Problem Solving" presents DEDUCTREASONER [6], 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 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.

# SYSTEM ARCHITECTURE AND METHODOLOGY

We propose a T5 model to translate the tokenized Linear equation word problem into mathematical equation and Reinforced Learning for the fine tuning of T5 model which used gradient approximation algorithm for to train agents to learn policies that maximize rewards. In this case, the agent will be a T5 model that has been trained to generate solutions to Linear Equations word problems. The reward function will be designed to encourage the T5 model to generate accurate and efficient solutions to Linear Equations word problems.

## System Block Diagram

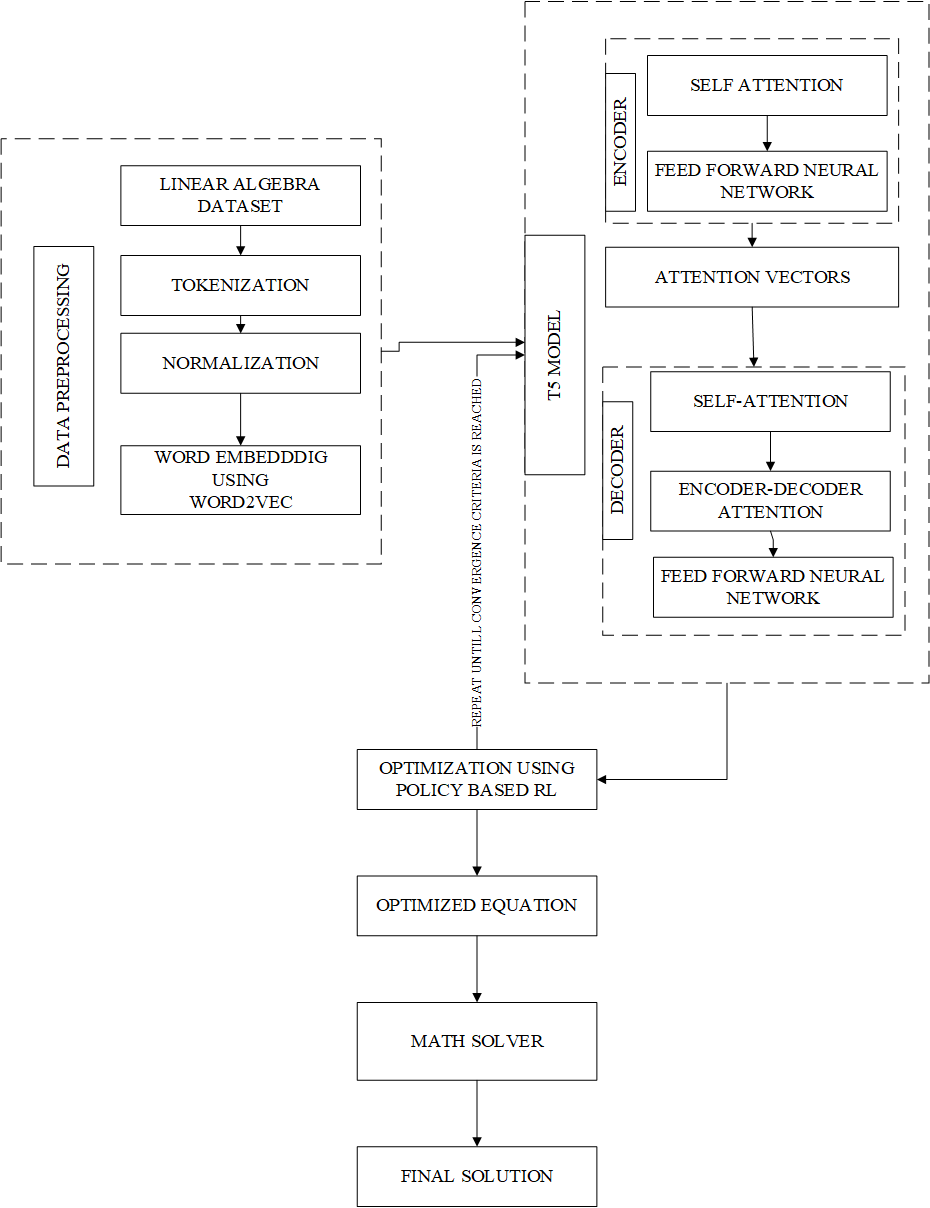


Figure 3 1: System Block Diagram of Linear Word Problem Solver

The proposed methodology is a promising approach for developing a Linear Equations word problem solver. The T5 model is a powerful language model that has been shown to be effective at a variety of tasks. Policy-based RL is a flexible and powerful reinforcement learning algorithm that can be used to train agents to learn complex tasks. By combining T5 and policy-based RL, we can develop a Linear Equations word problem solver that is able to generate accurate and efficient solutions to a wide range of problems.

## Dataset Processing

Dataset processing is an important step in our project Linear Equations word problem solver. Here are some of the detailed explanations of the data processing we will be following:

* **Data collection:** We will gather the algebraic word problems from various sources with all kinds of system to solve such as linear equations, quadratic equations or system of equations to ensure that the dataset covers a wide range of problem types, difficulty levels and includes both the problem statements and their corresponding solutions,
* **Data parsing:** After cleaning (into their canonical form) and making the problem statements and solutions are in a consistent format we will parse the problem statements and solution to extract relevant information. This step involves identifying the mathematical expressions, variables, numbers and other key element within problem statements. In this step we will leverage techniques like regular expression or NLP tools to tokenize and parse the text effectively.
* **Data Augmentation:** For the Linear Equation (Two Variable) Word Problem Solver, we will employ data augmentation techniques to enhance the dataset. Techniques such as numerical perturbations, variable substitutions, and structure-preserving transformations will be applied to generate diverse problem variations while maintaining the fundamental mathematical relationships.
* **Encoding and Tokenization:** We will convert the problem statements and solutions into a machine-readable format suitable for training with our chosen model (T5). We’ll tokenize the text by breaking it into smaller units like words or sub words, and then map these units to numerical representations using an appropriate vocabulary. We will ensure that the encoding scheme and tokenization process align with the requirements of T5.
* **Data Preprocessing and Normalization:** Depending on the specific requirements of our model and problem domain, we might need to preprocess the data further. This can include techniques like stemming or lemmatization for text normalization, handling missing values or outliers, scaling numerical values, or applying any domain-specific preprocessing steps that can enhance the model's performance.

## Transformer Architecture

The Transformer model architecture is a powerful neural network architecture that revolutionized various natural language processing tasks, including 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. In the context of your project, the Algebraic Math Word Problem Solver, the Transformer architecture can be used to process and understand the problem statements expressed in natural language.

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

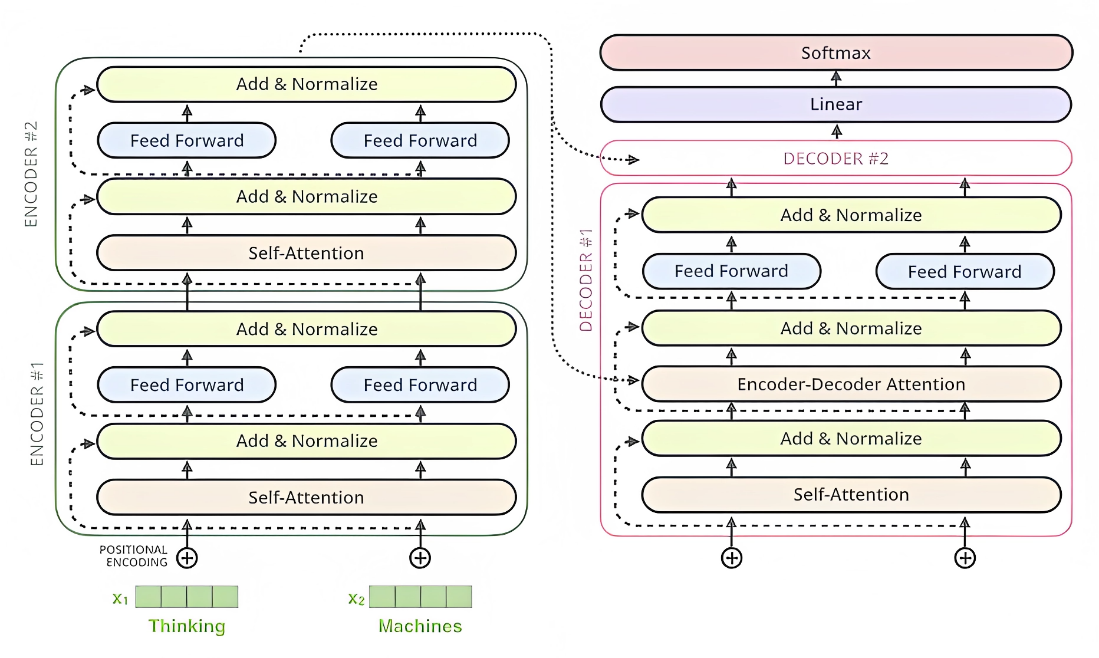


Figure 3 2: Transformer Architecture Model

### 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." Each token is initially embedded into a continuous vector space through an embedding layer.

### 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.

1. 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 we want to calculate the attention weights. 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.

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.

### 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.

### Feed-Forward Network

The output of the self-attention mechanism passes through a feed-forward neural network within each encoder. This network consists of two linear transformations separated by an activation function, typically a ReLU (Rectified Linear Unit). The feed-forward network introduces non-linear transformations to model complex interactions between words.

### Layer Normalization

After each sub-layer (self-attention and feed-forward network), layer normalization is applied to normalize the outputs. Layer normalization ensures that the model's outputs are more stable and facilitates faster convergence during training.

### Decoder Stack

The decoder stack is similar to the encoder stack but also includes an additional sub-layer of multi-head attention over the encoder's output. This allows the decoder to attend to the relevant parts of the input sequence when generating the solution for the algebraic math word problem.

### Output Layer

The final layer of the Transformer model is a linear transformation followed by a softmax activation function. It maps the model's representations to the vocabulary space and produces the probability distribution over possible output tokens. In the case of your project, the output layer generates the textual solution to the algebraic math word problem.

## 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 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.

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.

## T5 Model Training

For the training section. The preprocessed tokenized words and equations are the input to T5 model. The input and output texts are represented as sequences of tokens.

When mapping the input tokenized math word problem to equations, the tokenized problem is fed into the T5 encoder. The encoder processes the tokens and encodes them into a high-dimensional representation called the "context vector" or "hidden state." This context vector captures the semantic and contextual information of the input text.

Here are some steps that explains how context vector captures semantic and contextual information:

* Attention Mechanism: For each token in the input text, the attention mechanism calculates attention scores. The relevance or significance of each token in relation to the current decoding step is indicated by these scores.
* Attention Scores: The current state of the decoder and the encoder's hidden states are compared in order to calculate the attention scores. These scores show how the decoder and encoder tokens are aligned, and they tell you which parts of the input text should get more attention.
* Weighted Sum: The attention scores are normalized and used as weights to compute a weighted sum of the encoder hidden states. This weighted sum is the context vector, which captures the information from the relevant parts of the input text.
* Semantic and Contextual Information: The context vector contains a combination of information from the input text that is important for generating the output. By attending to specific tokens in the input, the model can capture the semantic meaning and contextual dependencies necessary for accurate decoding.
* The context vector from the encoder is then passed to the T5 decoder. The decoder takes the context vector as input and generates the output sequence, which represents the equation corresponding to the input math word problem. The decoder generates each token of the output sequence step by step, attending to the context vector and previously generated tokens to make predictions.
* During training, T5 is trained in a supervised manner using a dataset that pairs tokenized math word problems with their corresponding equations. The model is trained to minimize a loss function that measures the discrepancy between the predicted equations and the ground truth equations in the training data. This training process enables T5 to learn the mapping from input tokenized math word problems to equations.

In summary, T5 maps the input tokenized math word problems to equations by encoding the input in the encoder, generating the equation sequence in the decoder, and training the model to minimize the discrepancy between predicted and ground truth equations.

## Optimization

For the optimization and hyper-parameter tuning of T5 model, we can use the policy-based Reinforcement Learning. The role of Reinforcement Learning is to train T5 to get optimal accurate equations for a given math word problem.

The policy in Reinforcement learning can be defined in terms of π =(S,A,P,R)

Where S stands for the possible states, A for actions, P for probability and R for reward. For RL, we need to define the Environment, State, Action, and Reward.

These are the steps involved in this process

* **Define Environment:** The math word problem dataset is used as the environment for the RL agent. Each math word problem is considered as an episode.
* **Define State:** The state can be defined as the current math word problem.
* **Define Action:** The action is the generated equation or equation template by the T5 model.
* **Define Reward:** The reward is computed based on the correctness or quality of the generated equation compared to the ground truth equation. For example, if the generated equation is correct, a positive reward is given; otherwise, a negative reward is given.
* **Policy Optimization:** The RL agent optimizes the policy by updating the parameters of the T5 model based on the received rewards.

For this overall process, we need to design and define the RL agent which includes the specifying the RL agent architecture. In our case it would be based on policy-based methods. we need to apply following algorithm for parameter optimization using RL

1. **Initialize the RL agent:** Initialize the RL agent by setting up the policy network and initializing its parameters. For our case it can be T5 model
2. **Training Loop:**

* Select an example math word problem from the dataset
* Use the T5 model to generate equations.
* Evaluate the generated equation using predefined metric or reward function. This involves comparing generated equation with real equation.
* Update the policy: Use policy gradient methods to update the parameters of the T5 model based on the rewards obtained.
* Repeat steps 1 to 4 for a sufficient number of iterations or until convergence.

1. **Pseudocode:**

* Initialize T5 model with random parameters
* Define RL environment:
  + Input: Math word problem
  + Output: Generated solution
  + Reward: Based on correctness of the solution
* Choose a policy gradient algorithm (e.g., REINFORCE
* Initialize empty replay buffer
* for episode in range(num\_episodes):
* problem = sample\_math\_word\_problem()
* generated\_solution = T5\_model(problem)
* reward = evaluate\_solution(generated\_solution) # Evaluate the generated solution
* store\_experience(problem, generated\_solution, reward)
* if replay\_buffer.is\_full():
* for \_ in range(num\_iterations):
* batch = replay\_buffer.sample\_batch()
* compute\_gradients(batch)
* update\_model\_parameters()
* clear\_replay\_buffer()
* Save the trained T5 model

## Implementation Tools

### 3.7.1 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.

### 3.7.2 Software Requirements

* + **Python**: Python is a popular programming language widely used in the machine learning and NLP community. We'll need Python for implementing the algorithms, data preprocessing, model training, and evaluation.
  + **Deep Learning Frameworks**: A deep learning framework that supports transformer-based models like T5. Some commonly used frameworks include TensorFlow, PyTorch, and Hugging Face Transformers, which provide pre-implemented versions of T5.
  + **NLP Libraries**: Utilize NLP libraries such as NLTK, spaCy, or ransformers library from Hugging Face for text processing tasks like tokenization, parsing, and language modeling.
  + **Reinforcement Learning Libraries**: To incorporate policy-based RL, we'll need RL libraries such as OpenAI Gym, Stable Baselines, or Ray RLlib. These libraries provide implementations of RL algorithms and utilities for building RL environments and agents.
  + **Data Processing and Visualization Libraries**: Libraries like pandas, NumPy, and Matplotlib are essential for data preprocessing, analysis, and visualization tasks.

# EXPECTED OUTCOME

The expected result of the Linear Equation word problem solution employing the T5 model and policy-based RL for the given problem would be to present the user with a clear and simple answer. The solver would build the equations, optimize them using RL approaches, then use an equation solver library to get the solution. The intended outcome would be showing the final answers. The expected outcome is as following:

## Input problem into T5 model

Two numbers have a sum of 22. Their difference is 4. what are the two numbers?

## Output of t5 model

Assume the first number is 'X' and the second number is 'Y'

Equations:

X + Y = 22

X - Y = 4

## Reinforcement Learning (RL) optimization

The T5 model outputs the equations to the RL agent for optimization.

## RL-optimized equations

The RL agent suggests simplifying the equations by combining or rearranging them, if applicable.

## Equation Solver Library

The T5 model calls an external equation solver library to solve the optimized equations.

## Final steps and answers

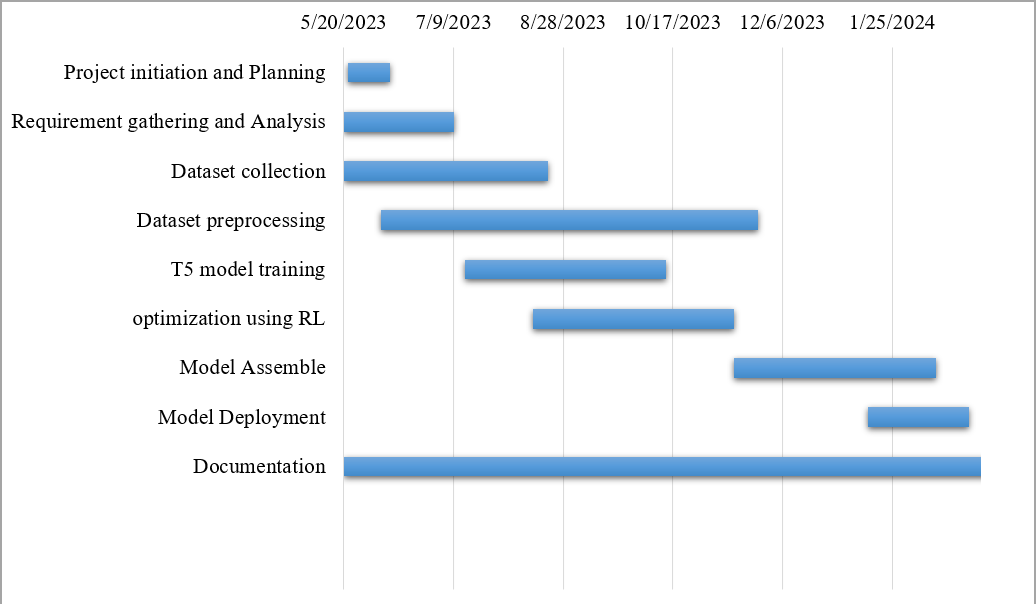
The T5 model incorporates the optimized equations and the solutions obtained from the equation solver library.

The final solution is obtained:

X = 13

Y = 9

# PROJECT SCHEDULE

Table 5 1: Gantt chart with Project Activities and Timeline

# FEASIBILITY STUDY

## 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.

## Economic Feasibility

The cost to develop this software would likely be relatively low compared to other types of software due to access to open-source tools and platforms.

## Technical Feasibility

The technical feasibility analysis focuses on the availability of appropriate technologies and resources needed for the solver's development and implementation. It was discovered that the required machine learning models, such as transformers and natural language processing approaches, are easily accessible. The initiative also guaranteed that appropriate computing infrastructure and resources were available to facilitate effective operation.

## 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.

# References

|  |  |
| --- | --- |
| [1] | S. B. Bussaba Amnueypornsakul, "Machine-guided Solution to Mathematical Word Problems," Department of Linguistics, Chulalongkorn University, 2014. |
| [2] | D. Z. J. Z. X. X. L. G. T. D. H. T. S. Lei Wang, "Template-Based Math Word Problem Solvers with Recursive Neural Networks," 2019. |
| [3] | Y. a. L. X. a. S. S. Wang, "Deep Neural Solver for Math Word Problems," Association for Computational Linguistics, 2017. |
| [4] | J. L. C.-Y. L. J. Y. Danqing Huang, "Neural Math Word Problem Solver with Reinforcement Learning," Association for Computational Linguistics, 2018. |
| [5] | Q. Z. Z. W. X. H. Qinzhuo Wu, "Math Word Problem Solving with Explicit Numerical Values," Association for Computational Linguistics, 2021. |
| [6] | J. L. W. L. Zhanming Jie, "Learning to Reason Deductively: Math Word Problem Solving as Complex Relation Extraction," 2022. |