

Leveraging Large Language Models for Bengali Math Word Problem Solving with Chain of Thought Reasoning

A Thesis

Submitted in partial fulfillment of the requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

Submitted by

Jalisha Jashim Era	20200104038
Bidyarthi Paul	20200104023
Mirazur Rahman Zim	20200104130
Tahmid Sattar Aothoi	20200104132

Supervised by

Mr. Faisal Muhammad Shah



Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Dhaka, Bangladesh

December 2024

CANDIDATES' DECLARATION

We, hereby, declare that the Thesis presented in this report is the outcome of the investigation performed by us under the supervision of Mr. Faisal Muhammad Shah, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE4100: Project and Thesis I and CSE4250: Project and Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this Thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Jalisha Jashim Era
20200104038

Bidyarthi Paul
20200104023

Mirazur Rahman Zim
20200104130

Tahmid Sattar Aothoi
20200104132

CERTIFICATION

This Thesis titled, “**Leveraging Large Language Models for Bengali Math Word Problem Solving with Chain of Thought Reasoning**”, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in December 2024.

Group Members:

Jalisha Jashim Era	20200104038
Bidyarthi Paul	20200104023
Mirazur Rahman Zim	20200104130
Tahmid Sattar Aothoi	20200104132

Mr. Faisal Muhammad Shah
Associate Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Prof. Dr. Md. Shamim Akhter
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

Our sincere gratitude goes to our distinguished supervisor, Mr. Faisal Muhammad Shah. Throughout this thesis, his constant encouragement, wise counsel, and constructive feedback have been invaluable. His extensive knowledge and dedication have greatly influenced the quality and direction of our study, enabling us to achieve meaningful results. We are deeply thankful for his visionary leadership, unwavering support, and the scholarly environment he fostered, which has significantly enriched our academic and research journey, making this work possible.

Dhaka
December 2024

Jalisha Jashim Era

Bidyarthi Paul

Mirazur Rahman Zim

Tahmid Sattar Aothoi

ABSTRACT

The task of solving Bengali Math Word Problems (MWPs) poses a significant challenge in natural language processing (NLP), especially for low-resource languages like Bengali. In this study, we leverage transformer-based models and large language models (LLMs) to address the complexities of Bengali MWPs. Two innovative datasets were introduced: BGSM8K, comprising 8,792 complex problems requiring multi-step reasoning, and PatiGonit, containing 10,000 simple arithmetic problems. Transformer-based models, including mT5, BanglaT5, and mBART50, were fine-tuned to generate accurate mathematical equations from Bengali text. Chain-of-Thought (CoT) prompting was employed to enhance reasoning capabilities in LLMs such as GPT-4, GPT-3.5 Turbo, Llama 3, 3.1, 3.2, and 3.3 achieving substantial improvements in solving complex MWPs. The mT5 model achieved the highest accuracy of 97.30%, highlighting its effectiveness in generating equation from MWPs. On the other hand, Gpt-4o achieved the highest accuracy of 83.80% for complex reasoning via Chain of Thoughts. Additionally, Low-Rank Adaptation (LoRA) was utilized for efficient fine-tuning, optimizing large models with minimal computational overhead. This work contributes significantly to Bengali NLP by providing robust frameworks, datasets, and methodologies for solving MWPs, bridging the resource gap for low-resource languages and paving the way for equitable educational technologies.

Contents

<i>CANDIDATES' DECLARATION</i>	i
<i>CERTIFICATION</i>	ii
<i>ACKNOWLEDGEMENT</i>	iii
<i>ABSTRACT</i>	iv
List of Figures	viii
List of Tables	ix
1 Introduction	1
1.1 Background	1
1.1.1 Chain-of-Thought Prompting	3
1.1.2 LoRA (Low-Rank Adaptation)	4
1.2 Problem Statement	5
1.3 Motivation	6
1.4 Objective	6
2 Related Work	8
2.1 Traditional Deep Learning Approaches	8
2.2 Transformer-Based Approaches	9
2.2.1 Hybrid and Ensemble Models	10
2.2.2 Methods Utilizing Intermediate Steps	10
2.2.3 Prompting-Based Approaches	11
2.2.4 Chain-of-Thought (CoT) Reasoning Approaches	12
3 Corpus Creation	14
3.1 Data Collection	14
3.1.1 BGSM8K Dataset	14
3.1.2 PatiGonit Dataset	15
3.2 Dataset Preparation	15
3.2.1 Data Translation	15

3.2.2	Translator Identity	16
3.2.3	Translation guidelines	16
3.2.4	Challenges during Translation	16
3.3	Dataset Statistics	17
3.3.1	BGSM8K Dataset	17
3.3.2	PatiGonit Dataset	17
4	Methodology	20
4.1	Evaluation measures	20
4.2	Proposed Approaches	21
4.2.1	Transformer Based Models	21
4.2.2	Large-Language Models	23
4.3	Prompting Techniques	25
4.3.1	Zero Shot Prompting	26
4.3.2	Few Shot Prompting	26
4.4	Fine Tuning	29
4.4.1	Gpt-3.5 Standard Tuning	29
4.4.2	LoRA Fine Tuning	30
5	Experimental Setup	32
5.1	Baseline Setup	32
5.2	Train-Test-Validation splits	32
5.3	Evaluation metrics	33
5.3.1	BGSM8K Evaluation	33
5.3.2	PatiGonit Evaluation	33
5.4	Hyperparameter Tuning	34
6	Result Analysis	36
6.1	Transformer based results	36
6.2	Large language Models (LLMs) based results	37
6.3	Fine Tuning based results	39
6.3.1	LoRA Fintune based results	39
6.3.2	Gpt-3.5 Finetune result	40
7	Conclusion and Future Work	42
8	Limitations	43
	References	44
A	Prompting Output	51

List of Figures

1.1	Bengali Math Word Problem Equation generation and solver example	2
1.2	Chain-of-Thought (CoT) Prompting enables LLMs (Large Language Models) to improve complex reasoning of MWPs (Math Word Problems)	3
1.3	Comparison of Standard fine-tuning and LoRA (Low-Rank Adaptation) for model training	5
3.1	Pipeline of BGSM8K and PatiGonit Development	14
3.2	Challenges faced during the translation of BGSM8K dataset	17
3.3	Challenges faced during the translation of PatiGonit dataset	18
3.4	Sample of BGSM8K Dataset	19
3.5	Sample of PatiGonit dataset	19
4.1	Schematic diagram of the Bengali Math word problem solver	21
4.2	Schematic diagram of our proposed prompting approach	24
4.3	Instruction (Prompt-1) for BGSM8K Dataset (Zero Shot)	26
4.4	Instruction (Prompt-1) for BGSM8K Dataset (Few Shot)	27
4.5	Instruction (Prompt-1) for PatiGonit Dataset (Few Shot)	28
4.6	Structure of JSON file format	30
4.7	LoRA Fine-Tuning Process: During and After Training (LoRA, 2024) [1] . . .	30
5.1	Bleu Score examples	33
A.1	Example of correct chains of thought produced by the GPT-3.5 for the BGSM8K dataset for Few Shot	52
A.2	Example of correct chains of thought produced by the GPT-4o for the BGSM8K dataset for Few Shot	53
A.3	Example of correct chains of thought produced by the GPT-3.5 for the BGSM8K dataset for Fine Tuning	54

List of Tables

2.1	A Summary of Existing Works in solving Math Word Problems	13
3.1	Detailed Information of Translators	16
3.2	Summary of Equation Types in our proposed dataset	18
4.1	Architectural comparison of Transformer and Traditional Deep Learning Models	22
5.1	Hyper Parameters	34
6.1	Performance of the proposed models on "PatiGonit" dataset	36
6.2	Chain of Thoughts prompting outperforms Standard prompting for various Large language models for the "BGSM8K" dataset. A benchmark has been set for the "PatiGonit" dataset using Few Shot prompting technique. All metrics are Accuracy (%). (*) represents that the accuracy is correct but the responses of the model was in English.	37
6.3	Ablation Study with three variations of LoRA finetuning on GSM8K dataset. All metrics are Accuracy (%).	39
6.4	Performance comparison of finetuned Gpt-3.5 with Gpt-4o. All metrics are Accuracy (%).	40

Chapter 1

Introduction

The challenge of solving natural language math word problems (MWPs) has intrigued researchers since the inception of artificial intelligence. These problems, often encountered by elementary students, require the conversion of textual descriptions into mathematical equations involving basic arithmetic operations such as addition, subtraction, multiplication, and division or some requires reasoning steps to get the correct answer. Automating this process using advanced natural language processing (NLP) techniques can significantly enhance educational tools.

1.1 Background

In this era, Question Answering (QA) systems are vital in Natural Language Processing (NLP). They are designed to understand and respond to user queries in a human-like manner, crucial for applications like search engines and virtual assistants (Yigit & Amasyali, 2023 [2]; Wu et al., 2020 [3]). Improving these systems are essential due to the growing demand for quick and accurate information retrieval. The structure of QA systems varies with the domain and question types (Rogers et al., 2021 [4]; Yigit & Amasyali, 2019 [5]), with Math Word Problems (MWPs) being a notable subtype.

MWPs pose significant challenges in QA systems (Xie & Sun, 2019 [6]; Zhang et al., 2020a [7]; Liang et al., 2021b [8]; Wang et al., 2019 [9]; Zhang et al., 2020b [10]; Shen & Jin, 2020a [11]), requiring more than basic pattern recognition. They involve understanding mathematical operators, quantities, and their relationships to produce a solution equation. This process entails identifying numerical values, selecting appropriate operations, and forming mathematical expressions with unknown variables. Efforts to teach computers to solve MWPs date back to the 1960s, with researchers employing rule-based methods to mimic human problem-solving strategies (Feigenbaum, 2003 [12]). Recent advancements

in NLP and machine learning have significantly improved the ability to tackle MWPs, particularly for simpler problems.

With recent technological advancements, we have made significant progress, particularly for simpler problems. Despite advancements in NLP and machine learning, there has been a notable gap in applying these technologies to the Bengali language. Bengali, being a low-resource language, lacks substantial computational resources and datasets to support extensive research and development in NLP applications. Addressing this gap, our study introduces an innovative approach to solving Bengali MWPs using transformer-based models, including the Basic Transformer (Vaswani et al., 2017) [13], mT5 (Xue et al., 2020) [14], BanglaT5 (Bhattacharjee et al., 2022) [15], and mBART50 (Tang et al., 2020) [16]. These models have been trained on a specifically curated dataset, "PatiGonit" means "Arithmetic" or "Elementary Mathematics".

In our study, we employed transformer-based models to identify mathematical equations embedded within the text of Bengali word problems (Nayak et al., 2019) [17]. We fine-tuned key hyperparameters, including learning rate, number of epochs, and batch size, to optimize model performance. Our approach involves predicting the equation from the word problem using deep learning-based natural language processing techniques. Once the equation is predicted, we use an equation solver to obtain the final answer. An example of the type of math word problem handled by our work is shown in Figure 1.1. The intermediate equations generated by the models are processed by our equation solver to produce the final solution.

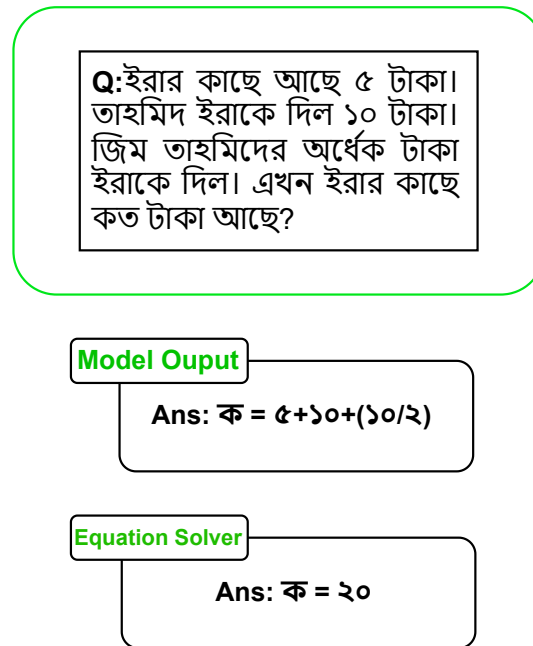


Figure 1.1: Bengali Math Word Problem Equation generation and solver example

1.1.1 Chain-of-Thought Prompting

While machine learning and transformer-based models excel at solving simpler MWPs (Nayak et al., 2019) [17], they often struggle with problems that require reasoning and multi-step solutions (Cobbe et al., 2021) [18]. These MWPs demand understanding complex relationships between quantities and applying mathematical principles sequentially, which many models fail to handle effectively. Language models perform well in NLP tasks, but their reasoning abilities are limited, which cannot be solved by increasing model size (Rae et al., 2021) [19]. Prompting-based tools fix this with QA tasks. Wei et al., 2022 [20] introduced CoT (Chain-of-Thought) prompting, which generates short sentences that mimic human problem-solving.

When solving a complex reasoning task, such as a multi-step math word problem, it is natural to break the problem into smaller intermediate steps and solve each one sequentially before arriving at the final answer. For instance: “After Era gives 2 lichies to her mom, she has 10 left. Then, after giving 3 to her dad, she has 7 left. So, the answer is 7.” A chain of thought is a series of intermediate natural language reasoning steps that lead to the final output, and we refer to this approach as chain-of-thought prompting (Wei et al., 2022 [20]). Recent advancements on chain-of-thought prompting, have shown promise in enhancing the reasoning capabilities of large language models for solving complex MWPs. Chain-of-thought prompting involves breaking down a problem into intermediate reasoning steps, which are then solved sequentially to reach the final answer.

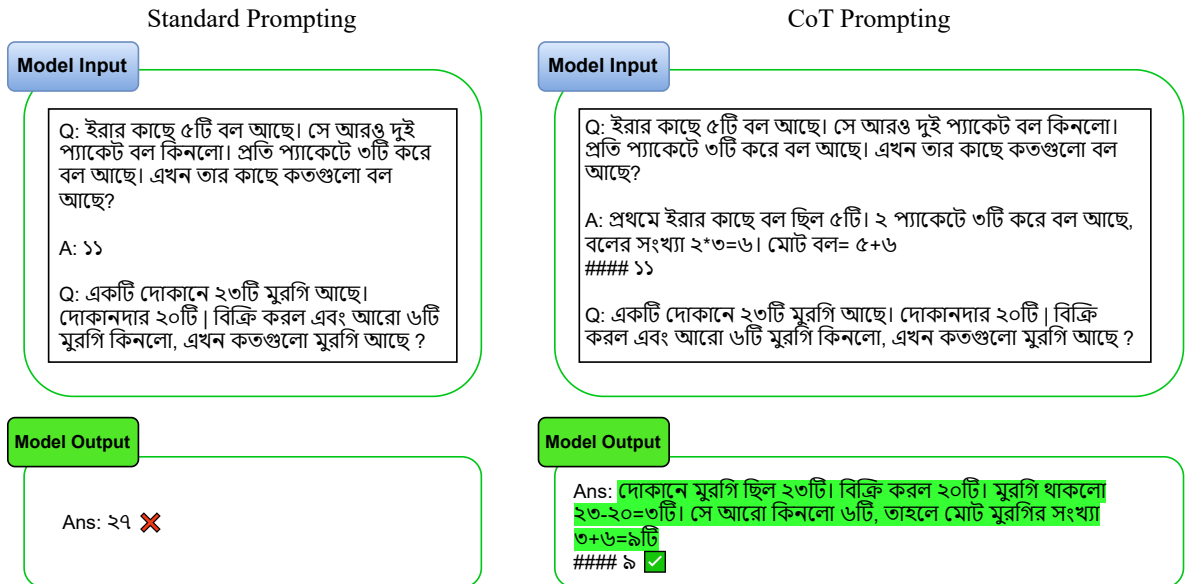


Figure 1.2: Chain-of-Thought (CoT) Prompting enables LLMs (Large Language Models) to improve complex reasoning of MWPs (Math Word Problems)

A comparison between Standard Prompting and Chain-of-Thought (CoT) Prompting for

solving math word problems (MWP) using a language model is shown in Figure 1.2. In Standard Prompting, the model is presented with the question directly, without any reasoning steps, leading to incorrect answers due to a lack of logical breakdown. In contrast, CoT Prompting provides a step-by-step explanation, allowing the model to reason through the problem logically and arrive at the correct solution. By breaking the problem into smaller parts and solving it systematically, CoT significantly improves the model's ability to handle complex reasoning tasks, as demonstrated by the accurate outputs in the CoT example.

Studies (Wei et al., 2022) [20] have demonstrated that incorporating chain-of-thought reasoning into few-shot prompting significantly improves model performance on tasks such as math word problems, commonsense reasoning, and symbolic manipulation. By leveraging this method, large language models can better handle the intricate reasoning required for complex MWPs.

The challenge is even greater for low-resource languages like Bengali, where limited datasets and linguistic complexities further hinder the development of reasoning-based solutions. Bengali is a low-resource language for Math Word Problem (MWP) solving using Large Language Models, with 272.7 million speakers and seventh worldwide [21]. Bengali language processing tools and models are difficult to develop due to a lack of datasets, linguistic resources, standardization, and dialectal variations, which hinders LLMs for MWPs solving.

This study improves the reasoning capabilities of large language models (LLMs) in solving Bengali Math Word Problems (MWPs) using two key ideas. Firstly, mathematical reasoning can be improved by natural language explanations for the final answer. By training from scratch or fine-tuning a pretrained model, models can generate intermediate steps (Ling et al., 2017 [22]; Patel et al., 2021 [23]). Neuro-symbolic methods use formal languages instead of natural ones (Roy & Roth, 2016 [24]; Chiang & Chen, 2018 [25]; Amini et al., 2019 [26]; Chen et al., 2019 [27]). Second, LLMs can prompt in-context few-shot learning by providing a few task examples instead of fine-tuning for each new task (Brown et al., 2020) [28]. This work uses chain-of-thought prompting to combine their strengths. The method includes prompts like input problem, reasoning steps, and output.

1.1.2 LoRA (Low-Rank Adaptation)

LoRA technique designed to optimize the fine-tuning of Large Language Models (LLMs), focusing on reducing memory usage and enhancing efficiency without compromising performance in Natural Language Processing (NLP) tasks.

LoRA (Hu et al., 2021) [29] is a parameter-efficient fine-tuning method that introduces new trainable parameters to modify a model's behavior without increasing its overall size. By doing so, LoRA maintains the original parameter count, reducing the memory overhead

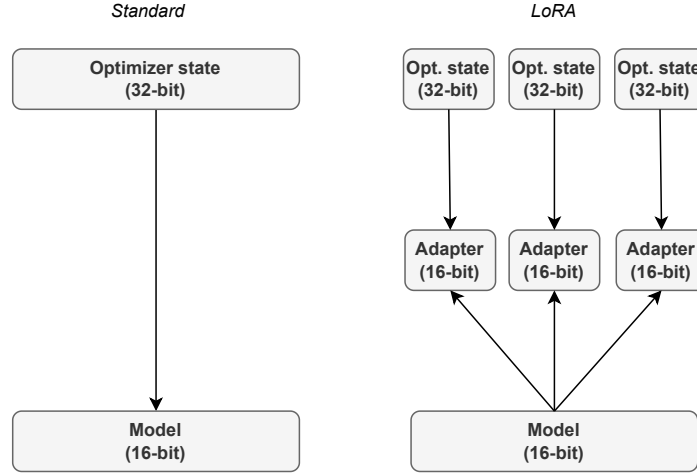


Figure 1.3: Comparison of Standard fine-tuning and LoRA (Low-Rank Adaptation) for model training

typically associated with training large models. It works by adding low-rank matrix adaptations to the model's existing layers, allowing for significant performance improvements while keeping resource consumption in check.

Figure 1.3 compares Standard fine-tuning with LoRA (Low-Rank Adaptation) for model training. In standard fine-tuning, the optimizer state (32-bit) directly updates the entire model (16-bit), which requires significant memory and computational resources. In contrast, LoRA introduces small, low-rank adapters (16-bit) that are updated instead of the full model, with the optimizer state (32-bit) applied to these adapters. This approach significantly reduces memory usage and computational overhead, as the base model remains frozen while only lightweight adapters are fine-tuned. LoRA thus provides an efficient alternative for model adaptation.

1.2 Problem Statement

First part of the problem can be framed as the generation of an equation and its corresponding answer by the model. In this case, the model generates the equation that represents the mathematical reasoning for solving the problem, followed by the predicted final result.

Let x denote the input math word problem, and $f(x)$ denote the output generated by the model, consisting of a mathematical equation e and a final result r_{pred} , where r_{true} is the ground truth final result for the problem.

The model's output can be represented as:

$$f(x) = (e, r_{\text{pred}})$$

where e is the equation generated by the model, and r_{pred} is the predicted final result.

Second part of the problem is best framed as a model’s ability to solve complex math word problems by generating intermediate reasoning steps and a final result. Evaluation focuses solely on the correctness of the final result, disregarding intermediate steps.

Let, x denotes the input math word problem, $f(x)$ denotes the output generated by the model, consisting of a sequence of intermediate reasoning steps followed by a final result r_{pred} , and r_{true} denotes the ground truth final result for the problem.

The model’s output can be represented as:

$$f(x) = (s_1, s_2, \dots, s_k, r_{\text{pred}})$$

where s_1, s_2, \dots, s_k are the intermediate steps and r_{pred} is the predicted final result.

1.3 Motivation

The motivation behind this research stems from the need to address the lack of computational resources and research in solving math word problems (MWPs) for low-resource languages like Bengali. While significant progress has been made in natural language processing (NLP) for high-resource languages, Bengali, spoken by millions, remains underserved despite its educational and cultural significance. MWPs, which challenge learners to convert textual descriptions into mathematical equations, are crucial for developing critical thinking and problem-solving skills. Automating this process using advanced NLP techniques can revolutionize educational tools, making them more accessible and effective for Bengali-speaking students. By focusing on Bengali MWPs, this study aims to bridge the resource gap, enhance multilingual NLP applications, and contribute to the development of equitable and inclusive educational technologies.

1.4 Objective

The following are the main objectives of our study:

- Creation of the “PatiGonit” dataset, consisting of 10,000 Bengali Math Word Problems from elementary school.
- Development of BGSM8k dataset containing 8792 complex Bengali Math Word Problems.

- Application and Fine-tuning of Transformer-based Models, including Basic Transformer, mT5, BanglaT5, and mBART50, specifically for solving Bengali Math Word Problems.
- Introduction of a pipeline for applying chain-of-thought prompting using the BGSM8k dataset, enabling large language models to handle complex reasoning tasks effectively.
- Introduction of two Prompts to solve Bangla MWPs, one for each dataset.
- Comparative analysis of performance for GPT-4o, GPT-3.5 Turbo-16k, Llama-3 8B and 70B, llama-3.1-70b-versatile, and llama-3.2-90b-text-preview, demonstrating the strengths of zero-shot prompting and fine-tuning to solve MWPs.
- Introduction of Low-Rank Adaptation (LoRA) to efficiently fine-tune large language models for Bengali math word problems, optimizing performance with minimal computational overhead.
- Establishment of robust frameworks for developing multilingual and interpretable math word problem solvers, focusing on low-resource languages like Bengali.

In summary, we introduced two new datasets, “BGSM8k”, which focuses on complex Bengali math word problems and includes intermediate reasoning steps (chain of thought) for each solution and “PatiGonit”, which concentrates on simple MWPs and includes equations with their corresponding answers as solutions. These two datasets highlight the limitations of traditional language models, which can only solve straightforward math problems but struggle with complex ones requiring reasoning. Our proposed pipeline for Chain-of-Thought prompting, built on the BGSM8k dataset, addresses this challenge by enabling large language models to tackle more reasoning-intensive tasks. Several large language models, including GPT-4o [30], GPT-3.5 Turbo-16k [31], Llama-3 8B and 70B [32], llama-3.1-70b-versatile [32], llama-3.2-90b-text-preview [32], and llama-3.3-70b-versatile [32] were evaluated for their ability to generate intermediate steps and equations. Chain-of-thought prompting has consistently outperformed standard prompting on arithmetic reasoning benchmarks. While GPT-4o excelled in few-shot learning across both datasets, it underperformed compared to the fine-tuned GPT-3.5 Turbo, which benefited from targeted optimization. To further improve model performance, we employed LoRA (Low-Rank Adaptation) fine-tuning. This method enables efficient model adaptation by modifying only a small subset of parameters, making it computationally and memory efficient. LoRA was necessary for our work to fine-tune large models like Llama-3 8B for the specific structure of Bengali math word problems while preserving their versatility for other tasks.

Chapter 2

Related Work

In this section, we discuss works related to our study. During our exploration, we found that no prior work has focused on Bengali Math Word Problems (MWP). However, there have been several studies on solving MWPs in English, which we categorize into six main approaches: Traditional Deep Learning Approaches, Transformer-Based Approaches, Hybrid and Ensemble Models, Methods utilizing intermediate steps, Prompting-based approaches, and Chain-of-Thought (CoT) Reasoning Approaches. Each of these categories is discussed in the following sections.

2.1 Traditional Deep Learning Approaches

These methods are increasingly being used to reduce manual effort and to enhance the performance of MWP solvers. One notable method involves using Sequence-to-Sequence (Seq2Seq) models to improve MWP solvers, Wang et al., 2017 [33] proposed a Seq2Seq RNN model to solve math word problems (MWPs), integrating a GRU-based encoder, LSTM-based decoder, and a similarity-based retrieval mechanism for improved accuracy. While the model achieved 58.1% accuracy on Math23k and 16.1% on Alg514, the hybrid approach boosted performance to 64.7% and 70.1%, respectively. However, it struggled with diverse datasets and failed to generate novel equation templates. To enhance accuracy, Wang et al., 2018 [34] introduced equation normalization and explored Seq2Seq architectures, including BiLSTM, ConvS2S, and Transformer models. Their ensemble approach achieved 68.4% accuracy on Math23k, outperforming individual models (66.7% for BiLSTM, 64.2% for ConvS2S, and 62.3% for Transformer), emphasizing normalization and model diversity. Zhang et al., 2016 [35] developed the Variational Neural Machine Translation (VNMT) model with latent variables to enhance semantic understanding, achieving BLEU scores of 32.07 (Chinese-English) and 19.58 (English-German), outperforming traditional models but

facing challenges with long sentences.

Huang et al., 2018 [36] improved Seq2Seq models with Copy and Alignment mechanisms, further optimized via Reinforcement Learning (RL), achieving accuracies of 44.5% on Alg514, 64.0% on NumWord, and 23.3% on Dolphin18KT6. Their hybrid model significantly boosted performance to 82.5%, 65.8%, and 33.2%, respectively. Zhang et al., 2020 [37] introduced the Graph-to-Tree (Graph2Tree) model, using graph-based encoders and tree-based decoders to effectively capture numerical relationships. It achieved accuracies of 83.7% on MAWPS and 77.4% on Math23k, addressing prior gaps in quantity reasoning. Similarly, Li et al., 2020 [38] utilized BiGraphSAGE encoders and tree-based decoders in their Graph2Tree model, achieving 78.8% on MAWPS and 69.65% on MathQA but faced challenges in fully incorporating structural reasoning.

Shen & Jin, 2020b [39] proposed a multi-encoder, multi-decoder framework combining sequence-based and graph-based encoders with sequence and tree decoders. This approach improved equation generation and achieved 78.4% on the Math23k test set but still had gaps in leveraging textual structure. Liang et al., 2021c [40] introduced MWP-BERT, a numeracy-augmented pre-trained language model tailored for MWPs, achieving state-of-the-art results of 84.7% on Math23k, 76.2% on MathQA, and 91.2% on Ape-clean. Devlin et al., 2018 [41] presented BERT, a bidirectional pre-trained model for language understanding, achieving 93.2% on SQuAD v1.1 and 83.1% on SQuAD v2.0. Liu et al., 2019 [42] enhanced BERT with RoBERTa by removing Next Sentence Prediction (NSP), applying dynamic masking, and using larger datasets, achieving state-of-the-art results like 90.2% on MNLI and 96.4

Radford et al., 2019 [43] introduced GPT-2, a generative pre-trained transformer excelling in zero-shot learning tasks, with notable performance on datasets such as LAMBADA (63.2%). Clark et al., 2020 [44] presented RuleTakers for reasoning over natural language rules, achieving 99% on synthetic datasets and over 90% on hand-authored rules, though paraphrased reasoning remained a challenge. Saeed et al., 2021 [45] extended this work with RuleBERT, fine-tuning RoBERTa for probabilistic logic reasoning, achieving up to 99% accuracy on overlapping rules. Finally, Liang et al., 2021a [46] proposed EVR, a T5-based model for explainable reasoning, excelling in multi-hop tasks with interpretable steps, achieving 97.0% on DU5 and up to 98.1% on Birds-Electricity tasks. These advancements demonstrate significant progress in NLP for MWPs, reasoning, and language understanding across diverse datasets.

2.2 Transformer-Based Approaches

Liang et al. [47] revisited MWP-BERT, highlighting its use of numeracy-augmented pre-training for math word problem solving. By integrating numerical reasoning and sym-

bolic placeholders, the model addressed gaps in capturing numerical variations in context. MWP-BERT achieved accuracies of 84.7% on Math23k, 76.2% on MathQA, and 91.2% on Ape-clean. The datasets used included Math23k, MathQA, and Ape-clean (81,225 filtered problems from Ape210K). Raiyan et al. [48] introduced a novel approach for improving MWP solvers by generating linguistic variants of problem statements using GPT-3. These paraphrased problems were solved using a DeBERTa-based solver, with majority voting employed for final predictions. This approach addressed the lack of robustness in existing models to paraphrased problems. The method achieved accuracies of 91.0% on MAWPS, 79.1% on PARAMAWPS, and 63.5% on SVAMP. The datasets included MAWPS, PARAMAWPS, and SVAMP.

2.2.1 Hybrid and Ensemble Models

Hybrid models that combine different approaches have shown significant improvements. An ensemble model by Wang et al. [49] utilizing BiLSTM and LSTM with equation normalization achieved 69.2% accuracy on the Math23K dataset. In the paper Xie and Sun [50] the GTS model, integrating GRU, TreeDecoder, and gated feedforward networks, achieved 74.3% on Math23K and 83.5% on the MAWPS single operation dataset. WARM has been introduced in Chatterjee et al. [51] and it uses a weakly supervised approach with a bidirectional GRU encoder and three fully connected networks as the decoder, achieving 66.9% on All Arith and 56.0% on Math23K.

2.2.2 Methods Utilizing Intermediate Steps

Numerous studies have demonstrated the many benefits of teaching neural networks to produce intermediate steps through fine-tuning or training. Natural language intermediate steps. Chen et al., 2022 [52] introduced the "Rationalize-Then-Predict" framework, a two-stage method aimed at improving model robustness in adversarial contexts by using rationalizers to extract relevant inputs before prediction. Although models like VIB and SPECTRA showed promise (e.g., 82.6% accuracy on FEVER), they remained vulnerable to strong attacks. Hancock et al., 2018 [53] proposed BabbleLabble, a weak supervision approach leveraging natural language explanations and semantic parsers, achieving an F1 score of 50.1 on the Spouse dataset but facing challenges with noisy labels and generalization. Similarly, Dua et al., 2020 [54] highlighted the utility of intermediate annotations in reading comprehension, reporting 85% accuracy with Random Forest but acknowledging limited dataset diversity. Zaidan et al., 2007 [55] explored annotator rationales, improving SVM performance to 92.2% on the Polarity Dataset but identified issues with annotation quality and domain applicability. Yao et al., 2021 [56] developed REMOTE, which incorporated

human-provided explanations to enhance language models, achieving 62.0% on HatEval and 92.7% on AmazonMusic, but faced scalability issues with labeled data. Hase & Bansal, 2021 [57] evaluated the utility of explanations for improving task performance, achieving 91.41% on e-SNLI while noting inconsistent benefits across tasks. Gu et al., 2022 [58] introduced DREAM, which refined internal representations in QA models like Macaw, improving accuracy by 4% on CODAH but exposing limitations in coherent scene modeling. Andreas et al., 2017 [59] proposed Learning with Latent Language (L3) to parameterize multitask scenarios, showing promise in structured tasks like ShapeWorld (70% accuracy) but struggling with generalizing to abstract concepts. These studies collectively emphasize advancements in integrating explanations and rationales into AI systems while highlighting persistent challenges in scalability, robustness, and generalization.

Camburu et al., 2018 [60] introduced the e-SNLI dataset, a natural language inference (NLI) benchmark that incorporates explanations, enabling models to predict decisions and justify them. Models like e-INFERSENT achieved 83.96% accuracy, while EXPLAIN THE PREDICT ATTENTION scored 81.71%, though reliance on spurious correlations in existing models revealed gaps in robustness. Rajani et al., 2019 [61] developed the CoS-E dataset and CAGE framework for commonsense reasoning, which improved model accuracy to 72.6% on the CoS-E dev split but highlighted limitations in explanation generation and dataset availability, underscoring the importance of leveraging annotated explanations for better interpretability. Zelikman et al., 2022 [62] proposed the Self-Taught Reasoner (STaR), which combines reasoning and rationalization with models like GPT-J and BERT, achieving notable accuracies, such as 72.6% on CommonsenseQA and GSM8K. However, STaR faced challenges in scalability, requiring significant resources for rationale generation and balancing the trade-off between accuracy and explainability. Collectively, these works highlight the potential of explanation-based approaches in enhancing transparency, reasoning, and trust in AI systems while identifying gaps in dataset availability and model generalization.

2.2.3 Prompting-Based Approaches

Recent advancements in prompting techniques have led to significant improvements in the performance of language models. Building on the concept of few-shot prompting introduced by Brown et al., 2020 [28], various methods have been explored to refine and optimize how models interact with input data. For instance, Lester et al., 2021 [63] proposed automatic prompt learning techniques, enabling models to adapt their prompts based on the task at hand, rather than relying solely on predefined prompts. In addition, recent works such as Wei et al., 2021 [64], Sanh et al., 2021 [65], and Ouyang et al., 2022 [66] have focused on providing models with task-specific instructions, allowing for better alignment with the desired output.

Another significant development in this area is the use of detailed task instructions, which has been shown to further enhance the performance of language models. Raffel et al., 2020 [67], Wei et al., 2021 [64], Ouyang et al., 2022 [66], Sanh et al., 2021 [65], and Wang et al., 2022b [68] emphasize that enriching input-output pairs with clear and specific guidance helps improve model understanding and reasoning, especially in complex tasks like math word problems. This trend of augmenting prompts with additional context and instructions has been pivotal in achieving state-of-the-art results in various natural language processing tasks, demonstrating the importance of prompt design in optimizing model behavior.

2.2.4 Chain-of-Thought (CoT) Reasoning Approaches

Chain of Thought (CoT) prompting has become a widely adopted method for improving the performance of language models, especially in tasks requiring multi-step reasoning. By breaking down a complex problem into smaller, more manageable steps, CoT helps models generate clearer and more accurate answers. Wei et al., 2022 [20] demonstrated that CoT prompting significantly outperforms traditional methods for solving math word problems (MWPs), enhancing both the accuracy of the results and the transparency of the model's reasoning process.

Building on this, Li et al., 2022b [69] introduced DIVERSE (Diverse Verifier on Reasoning Step), a framework designed to improve language model reasoning by generating diverse prompts. This approach explores multiple reasoning paths for the same question and uses a verifier to filter out incorrect answers through a weighted voting system. Importantly, DIVERSE focuses on evaluating each reasoning step individually rather than the entire chain, which further refines the model's decision-making process. In a related study, Wang et al., 2022a [70] introduced the concept of self-consistency, a decoding strategy that selects the most consistent answer from multiple reasoning paths, providing another layer of validation. When applied to arithmetic reasoning benchmarks, self-consistency has been shown to enhance CoT prompting, yielding more accurate and reliable results.

Our study explores methods for solving Math Word Problems (MWPs), categorized into Deep Learning, Transformer-based, Hybrid, Prompting, and Chain-of-Thought (CoT) Reasoning approaches. The most significant gap identified in this study is the complete absence of prior work addressing Math Word Problems (MWPs) in the Bengali language. This highlights a critical underrepresentation of Bengali in existing research on mathematical reasoning and natural language processing. However for english dataset, traditional models like Seq2Seq and Graph2Tree advanced MWP solving but struggled with diverse datasets and reasoning structures. Transformer-based models, including MWP-BERT and GPT-3, achieved state-of-the-art results but faced robustness issues. Prompting and CoT methods improved multi-step reasoning by leveraging task-specific instructions and diverse reasoning paths. A summary

of the existing works is given in Table 2.1.

Table 2.1: A Summary of Existing Works in solving Math Word Problems

Paper Title	Approach	Model	Dataset	Accuracy
Deep neural solver for math word problems [33]	Seq2Seq, Hybrid: Seq2Seq with Retrieval	GRU encoder, LSTM decoder, similarity-based retrieval, and hybrid model.	Math23k	Math23k: Seq2seq 58.1%, Hybrid 64.7%
Math word problem solving by generating linguistic variants of problem statements [48]	Enhanced Mask Decoder and a Voting Mechanism	DeBERTa	MAWPS, PARA-MAWPS	MAWPS: 91.0%, PARA-MAWPS: 79.1%
Graph-to-tree learning for solving math word problems [37]	Graph2Tree	Graph Transformer with GCNs, Tree-based Decoder (pre-order traversal)	MAWPS, Math23K	MAWPS: 83.7%, Math23K: 77.4%.
Chain-of-thought prompting elicits reasoning in large language models [20]	Chain-of-Thought Prompting	GPT-3, LaMDA, PaLM, UL2, Codex	GSM8K	GSM8K: 63.1%
Making Large Language Models Better Reasoners with Step-Aware Verifier [69]	Diverse Verifier on Reasoning Step	OpenAI Models: davinci, text-davinci-002, code-davinci-002	GSM8K	GSM8K: 83.2%
Self-consistency improves chain of thought reasoning in language models [70]	Self-Consistency, Chain-of-Thought Prompting (CoT):	GPT-3: code-davinci-001, code-davinci-002, LaMDA-137B, PaLM-540B	GSM8K	GSM8K: 83.2%

Chapter 3

Corpus Creation

3.1 Data Collection

As per our exploration, we found no publicly available Bengali Math Word Problem (MWP) dataset containing complex math problems and reasoning steps with solutions or containing simple equations with solutions. To address this gap, we developed two new Bengali math word problem datasets, “BGSM8K” and “PatiGonit”, in this work. BGSM8K focuses on problems that require advanced reasoning and step-by-step solutions, designed to facilitate the evaluation of large language models’ reasoning capabilities. On the other hand PatiGonit refers to the branch of mathematics dealing with basic numerical operations such as addition, subtraction, multiplication, and division. Figure 3.1 illustrates the pipeline we employed to develop the BGSM8K and PatiGonit datasets.

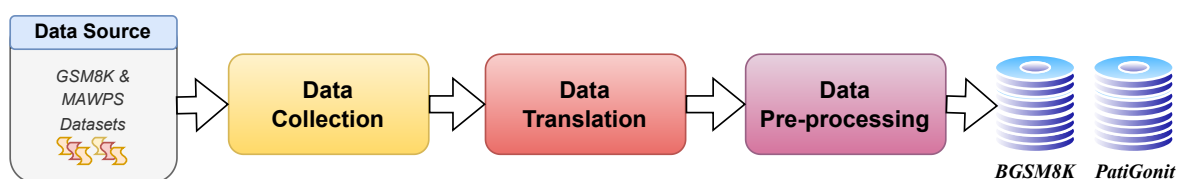


Figure 3.1: Pipeline of BGSM8K and PatiGonit Development

3.1.1 BGSM8K Dataset

To develop our Bengali math word problem dataset with intermediate steps and answers, we required a substantial collection of problems reflecting complex reasoning processes. As a foundation, we utilized the publicly available¹ GSM8K dataset [18], which contains

¹https://github.com/openai/grade-school-math/tree/master/grade_school_math/data

a diverse set of English grade school math problems. GSM8k has four parts: train, train socratic, test, and test socratic. But specifically, we leveraged the train and test portions of GSM8K, comprising a total of 8,792 questions and solutions. GSM8K was selected for its comprehensive coverage of grade school math concepts, making it an ideal base for developing our dataset. Inspired by GSM8K, we created BGSM8K (Bengali Grade School Math 8K). Here, 8K represents the more than 8 thousands total amount of data comprises in our dataset.

3.1.2 PatiGonit Dataset

For our second dataset, we introduced the Bengali Math Word Problems dataset named “PatiGonit”, comprising 10,000 math word problems. It traditionally refers to the branch of mathematics dealing with basic numerical operations such as addition, subtraction, multiplication, and division. These problems, initially in English within the MAWPS [23] dataset, which encompasses a total of 38,000 problems, underwent the translation process for the conversion into Bengali. Due to resource constraint we could only analyzed **10,000** of these problems. The dataset, typical of elementary school-level math, includes both basic arithmetic and some algebraic challenges, organized into two columns: one for the word problems and the other for the corresponding mathematical equations.

3.2 Dataset Preparation

3.2.1 Data Translation

In this section, we describe the dataset translation process for the BGSM8K and PatiGonit datasets. Manual data translation is inherently challenging as it requires not only linguistic accuracy but also the preservation of contextual meaning, particularly for reasoning-based tasks in math word problems. Any loss of reasoning structure during translation could compromise the dataset’s usability for downstream applications like reasoning evaluation and problem-solving.

To ensure the reasoning quality and contextual fidelity of the translated texts, we employed expert human translators for the manual translation process. Comprehensive guidelines were established to provide translators with clear instructions on preserving mathematical and contextual details. These predefined standards, combined with the translators’ expertise and judgment, ensured the integrity and consistency of the translated dataset.

3.2.2 Translator Indentity

For the translation process, we engaged professionals with expertise in English-to-Bengali translation to ensure that the translations were accurate, contextually consistent, and aligned with the reasoning requirements of the original texts. A team of five skilled individuals collaboratively worked on the translation task. The entire corpus of the BGSM8K dataset was evenly distributed among them to ensure efficiency and maintain uniform quality across the translated content. The information regarding their expertise, experiments and other details are presented in Table 3.1.

Table 3.1: Detailed Information of Translators

Details	Translator 1	Translator 2	Translator 3	Translator 4	Translator 5
Role	Graduate	Under-graduate	Under-graduate	Under-graduate	Under-graduate
Age	30	23	24	24	22
Research field	NLP	NLP	NLP	NLP	NLP
Experience	4 years	2 years	1 years	2 years	1 years

3.2.3 Translation guidelines

To ensure accuracy, cultural relevance, and linguistic diversity, the following guidelines were established and provided to the translators:

- Translations should retain the original meaning and reasoning of the text, ensuring that mathematical logic and contextual nuances are preserved.
- The Dollar symbol was consistently replaced with the Bengali Taka symbol to align with regional monetary conventions.
- English numerals were replaced with their Bengali counterparts to maintain consistency with the Bengali script.
- Individual names were replaced with culturally appropriate Bengali names.
- Similarly, Location names, objects, and food names were substituted with Bengali equivalents to ensure cultural relevance while maintaining the problem’s original structure and meaning.

3.2.4 Challenges during Translation

Even with the predefined guidelines, translators encountered several challenges during the translation process, which are summarized in Figure 3.2 and 3.3.

Word Problem in proposed BGSM8K Dataset	Word Problem in English	Challenges
মিরার স্থানীয় পিঠার দোকান ১০০টি ভিন্ন স্বাদের পিঠা প্রদান করে। তিনি ২ বছর আগে ১/৪ স্বাদের অভিজ্ঞতা নেয় এবং গত বছর সেই পরিমাণ দ্বিগুণ হয়। ১০০টি স্বাদের চেষ্টা করার জন্য তাকে এই বছর আরও কত ধরনের স্বাদ নিতে হবে?	Gretchen's local ice cream shop offers 100 different flavors. She was able to try a 1/4 of the flavors 2 years ago and double that amount last year. How many more flavors will she need to try this year to have tried all 100 flavors?	Understanding the pronoun of the proper noun
আইমা এবং জায়দা হকার মার্কেটে কিছু হাতের তৈরী জিনিস কিনতে গিয়েছিল। আইমা বাজারে জায়দার চেয়ে ২/৫ গুণ বেশি টাকা খরচ করেছে। জায়দা ৮৪০০ খরচ করলে, তারা একসাথে কত টাকা খরচ করেছে তা নির্ণয় করুন।	Aitana and Jayda went to Silverlake Flea market to buy some handmade crafts. Aitana spent 2/5 times more money than Jayda at the market. If Jayda spent \$400, calculate the total amount of money they spent together.	Ignorance of Irrelevant sentences
একটি সাপের মাথা তার দৈর্ঘ্যের এক দশমাংশ। যদি একটি সাপ ১০ ফুট লম্বা হয়, তবে তার মাথা বাদে শরীরের বাকি অংশের দৈর্ঘ্য নির্ণয় করুন।	A snake's head is one-tenth its length. If a snake is 10 feet long, calculate the length of the rest of its body minus the head.	Identify the simple sentence from the compound sentence
মুনতাহা সামির ১৯ টাকা এবং ৬ পয়সা রয়েছে। তিনি ৩ টাকায় ৪টি মিছরি এবং ১ পয়সা দিয়ে ১টি বেলুন কেনেন। পয়সায় তার কত টাকা বাকি আছে?	Sam has 19 dimes and 6 quarters. She buys 4 candy bars for 3 dimes each and 1 lollipop for 1 quarter. How much money, in cents, does she have left?	Understanding the given object key word to solve the problem
মোহিমার বয়স ৩ বছর। তার ভাইয়ের বয়স তার চেয়ে ৪ গুণ। তার বোন তার ভাইয়ের থেকে ৫ বছরের ছোট। মোহিমার এবং তার বোনের মধ্যে বয়সের পার্থক্য কী?	Mandy is 3 years old. Her brother is 4 times as old as she is. Her sister is 5 years younger than her brother. What is the age difference between Mandy and her sister?	Identification of the indirect object pronoun

Figure 3.2: Challenges faced during the translation of BGSM8K dataset

3.3 Dataset Statistics

3.3.1 BGSM8K Dataset

The BGSM8K dataset is composed of 8,792 Bengali math word problems with intermediate reasoning steps and their corresponding answers. A Sample is shown in Figure 3.4.

3.3.2 PatiGonit Dataset

We present an analysis of the equations contained within the dataset, categorizing them into simple and complex equations based on the number of mathematical symbols used. Simple

Word Problem in proposed BMAWPS Dataset	Word Problem in English	Challenges
লিজার ৩৪ টি প্রেস ছিল। টমাস তাকে আরো কিছু সরবরাহ করেছে। এখন লিজার ৬৪ টি প্রেস আছে। টমাস তাকে কতগুলো সরবরাহ করেছিল?	Liza had 34 Press. Thomas furnished him some more. Now Liza has 64 Press. How many did Thomas furnish him?	Understanding the pronoun of the proper noun
মেরি ১২৫ টি আম বন্ধুদের মধ্যে ২৫টি করে চকলেট দিতে চায়। প্রতিটি বন্ধু কতজন পাবে?	George wants to distribute 125 mangos among 25 friends. How many would each friend acquire?	Identify the simple sentence from the compound sentence
টিমির ৮৬টি তরমুজ ছিল। কান্দি তার কাছে থেকে ৪০ টি টাক নিয়ে। এখন কত তরমুজ টিমি কমেছে?	Timmy had 86 watermelons. Kandi took 40 from him. Now how many watermelons Timmy has decreased by?	Understanding the given object key word to solve the problem
উইলিয়ামের ৯৯টি নাশপাতি ছিল। মার্টিন তার কাছে থেকে ৬৭টি নাশপাতি নিয়ে। এখন উইলিয়ামের কতটি নাশপাতি বাকি আছে?	William had 99 pear. Martin took 67 from him. Now how many pear William have left over?	Identification of the indirect object pronoun

Figure 3.3: Challenges faced during the translation of PatiGonit dataset

equations are defined as those containing only one mathematical operation, while complex equations involve multiple operations. The Table 3.2 below summarizes the total count of simple and complex equations, as well as a further breakdown of the simple equations into four categories: addition, subtraction, multiplication, and division. Sample of the dataset is shown in Figure 3.5.

Table 3.2: Summary of Equation Types in our proposed dataset

Equation Types		Count
Simple Equations	Addition (+)	1761
	Subtraction (-)	3217
	Multiplication (*)	1610
	Division (/)	3164
Complex Equations	Mixed (+, -, *, /)	248

In summary, BGSM8K and PatiGonit datasets are designed to address Bengali math word problems, with BGSM8K containing 8,792 problems covering grade-school math concepts and PatiGonit offering 10,000 translated problems from the MAWPS dataset. Both datasets include problems with intermediate reasoning steps and corresponding answers. The translation process from English to Bengali involved expert human translators to ensure accuracy and contextual relevance. The datasets feature a range of simple and complex equations.

Question: রুমি এবং আরও ৪ জন মহিলা প্রতিদিন এক সাথে হাটে। সব মহিলারা একসাথে মোট ৩ মাইল হাটেন। তার উপরে, জেমি ৬ দিনের জন্য প্রতিদিন অতিরিক্ত ২ মাইল হাটে যখন তার বন্ধু সুমি ৬ দিনে সেই পরিমাণের অর্ধেক হাটে। যদি তারা সপ্তাহে ৬ দিন একই পথে হাটে, তাহলে মহিলারা মোট কত মাইল হাটবে? (Rumi and four other women walk together every day. Together, all the women walk a total of 3 miles. Additionally, Jamie walks an extra 2 miles each day for 6 days, while her friend Sumi walks half that amount over the same 6 days. If they walk on the same path 6 days a week, how many miles will the women walk in total during the week?)

Solution: দলে মোট ৫ জন মহিলা রয়েছেন। তারা সকলেই ৬ দিন প্রতিদিন ৩ মাইল হাটেন তাই $৩ \times ৬ = ১৮$ মাইল। রুমি ৬ দিনের প্রতিদিন অতিরিক্ত ২ মাইল হাটে তাই $২ \times ৬ = ১২$ মাইল। সুমি রুমির অর্ধেক দূরত্ব অতিক্রম করে তাই $১২/২ = ৬$ মাইল। একসাথে, মহিলারা $১৮ + ১২ + ৬ = ৩৬$ মাইল হাটেন। #### ৩৬ (There are a total of 5 women in the group. They all walk 3 miles every day for 6 days, so $3 \times 6 = 18$ miles. Jamie walks an additional 2 miles each day for 6 days, so $2 \times 6 = 12$ miles. Sumi covers half the distance that Jamie walks, so $12 \div 2 = 6$ miles. Altogether, the women walk $18 + 12 + 6 = 36$ miles. #### 36)

Figure 3.4: Sample of BGSM8K Dataset

Question: ক্যাসি ১৭ জন বন্ধুদের মধ্যে কিছু রুটি ভাগ করতে চায়। যদি প্রতিটি বন্ধু ৩৯টি রুটি পায়, তাহলে সব বন্ধু মোট কতটি রুটি পাবে? (Casey wants to share some Bread among 17 friends. If each friend gets 39 Bread, then how many Breads would every friend of his have?)

Equation: $k = ৩৯ \times ১৭$
($X = 39 * 17$)

(a) Simple Equation

Question: একটি বিন্ডিংয়ে ১২টি কক্ষ রং করার জন্য একজন চিত্রশিল্পীর প্রয়োজন হয়। প্রতিটি কক্ষ রং করতে ৭ ঘন্টা সময় লাগে। যদি সে আগে থেকেই ৫টি কক্ষ রং করে থাকে, তাহলে বাকি কক্ষ রং করতে তার আর কত সময় লাগবে? (A painter needed to paint 12 rooms in a building. Each room takes 7 hours to paint. If he already painted 5 rooms, how much longer will he take to paint the rest?)

Equation: $k = (৭.০ * (১২.০ - ৫.০))$
($X = (7.0 * (12.0 - 5.0))$)

(b) Complex Equation

Figure 3.5: Sample of PatiGonit dataset

Chapter 4

Methodology

Our methodology focuses on both transformer-based models and large language models (LLMs) for their effectiveness in handling Bengali math word problems. Specifically, we assessed transformer-based models such as MT5, BanglaT5, and mBART50 along with Basic Transformer, which were chosen over traditional deep learning architectures like LSTM and RNN due to their superior capability in managing complex sequence-to-sequence tasks. In addition, we explored the performance of LLMs by creating prompts, feeding them into the models via API configurations, and analyzing the outputs. This dual approach highlights the comparative strengths and suitability of both transformer-based models and LLMs for solving Bengali math word problems.

4.1 Evaluation measures

For the “PatiGonit” dataset we evaluated our models using two metrics: BLEU score and solution accuracy. The BLEU score measures the similarity between predicted and reference outputs, but it focuses on surface-level matching. This means it may not fully reflect correctness if a model generates the right equation with slight variations in token arrangement, potentially resulting in a lower score.

Accuracy was used as the sole metric for evaluating the “BGSM8K” dataset. In math word problems, the primary requirement is the correctness of the final answer, rather than the specific intermediate steps used to reach that answer. Therefore, other metrics were deemed unnecessary.

4.2 Proposed Approaches

A dual-methodology approach was explored for solving Bengali math word problems, leveraging both transformer-based models and large language models (LLMs) to assess their performance and suitability for this complex task.

4.2.1 Transformer Based Models

Transformers effectively manage long-range dependencies in text, crucial for solving math word problems. Unlike LSTMs and RNNs, which struggle with longer sequences, transformers process entire sequences in parallel, capturing context more efficiently for accurate equation generation. While BERT excels at understanding text, its encoder-only architecture makes it less suited for generative tasks like equation generation. By fine-tuning pretrained versions of mT5, BanglaT5, and mBART50 on our “PatiGonit” dataset, we optimized the models to excel at generating accurate mathematical equations from Bengali text. mT5 was trained on 101 languages, mBART50 on 50 languages, and BanglaT5 was specifically trained on Bengali. In Figure 6, this is illustrated by providing examples in English, Bengali, French, and Italian, demonstrating the models’ multilingual capabilities in understanding and processing text across different languages. Figure 4.1 represents the schematic diagram of the Bengali Math word problem solver.

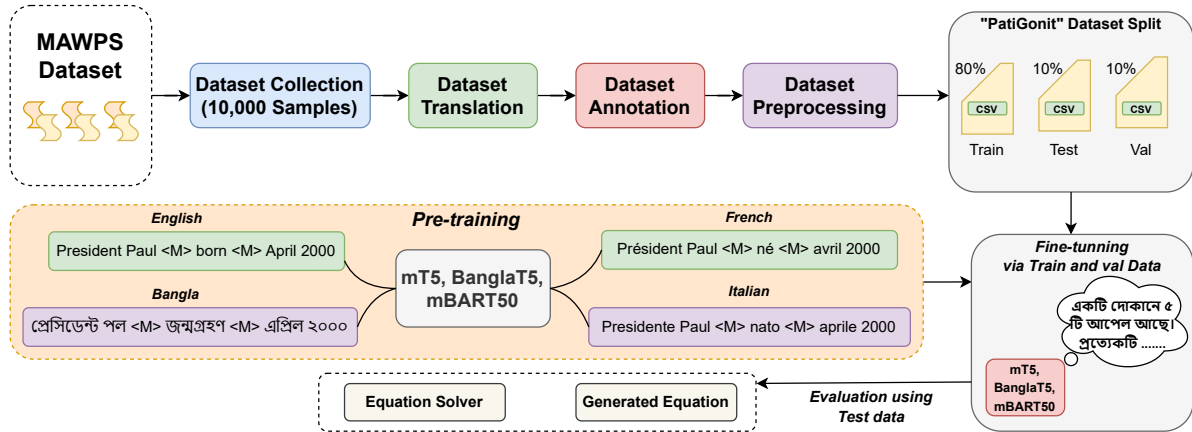


Figure 4.1: Schematic diagram of the Bengali Math word problem solver

To better understand the differences between the selected transformer models (MT5, BanglaT5, mBART50) and other deep learning architectures (BERT, LSTM, RNN), Table II provides a comparative analysis. This table highlights the variations in architecture, task type suitability, and training corpus, which illustrates why transformer-based models are better suited for our task of translating math word problems into mathematical equations. An overview of the models are given in Table 4.1.

Table 4.1: Architectural comparison of Transformer and Traditional Deep Learning Models

Feature	BERT	LSTM	RNN	mT5	BanglaT5	mBART50
Architecture	Encoder-only (bi-directional)	Recurrent, with memory cells	Recurrent, simpler than LSTM	Encoder-Decoder (Seq2Seq)	Encoder-Decoder (Seq2Seq)	Encoder-Decoder (Seq2Seq)
Task Type	Text classification, NLU	Sequential tasks (time series, NLP)	Sequential tasks (time series, NLP)	Text generation, NLG + NLU	Text generation, NLG + NLU	Text generation, NLG + NLU
Training Corpus	Wikipedia, BookCorpus, etc.	Task-specific	Task-specific	mC4 (multilingual)	Bengali language corpus	CC25, Multilingual (50 languages)

Basic Transformer

The complexity of math word problems poses a significant challenge in natural language processing, particularly in converting text to mathematical equations. To tackle this, we employed the Basic Transformer model, renowned for its success in sequence-to-sequence tasks. Developed by Vaswani et al., 2017 [13], the transformer architecture processes entire texts simultaneously, enhancing context comprehension, which is crucial for accurately generating corresponding equations from word problems. The attention mechanism of our transformer was followed from [17] and the attention function from that study is given below.

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (4.1)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (4.2)$$

$$\text{where, head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

mT5

The mT5 is a multilingual variant of the T5 model designed for handling a wide range of natural language processing tasks by framing them as text-to-text problems. It uses an encoder-decoder architecture, which allows it to understand input text and generate corresponding output text, making it highly effective for tasks like translation, summarization, and question answering. One of its key features is its ability to work across 101 languages, as it has been pretrained on a diverse multilingual dataset known as mC4. This makes mT5 particularly relevant for our work, as it can efficiently translate Bengali math word problems into mathematical equations. Additionally, a study [71] demonstrates the potential of mT5 in handling multilingual math word problems, highlighting its capability to generate accurate equations from textual descriptions [14]. Its strength in both understanding and generating text makes it a strong choice for solving word problems in low-resource languages like Bengali.

BanglaT5

BanglaT5 is a language-specific variant of the T5 model, specifically tailored for the Bengali language [15]. Like mT5, it uses an encoder-decoder architecture designed to handle text-to-text tasks, making it highly effective for tasks such as translation, summarization, and question answering. BanglaT5 is pretrained on a large Bengali corpus, allowing it to understand the linguistic details and contextual dependencies specific to the Bengali language. This makes it particularly suitable for our work, where the goal is to translate Bengali math word problems into corresponding mathematical equations.

mBART-50

mBART50 is a multilingual sequence-to-sequence transformer model, part of the mBART family, pretrained on data from 50 languages. Like mT5 and BanglaT5, mBART50 utilizes an encoder-decoder architecture, making it highly effective for tasks that involve both understanding and generating text. Its ability to perform translation, summarization, and other generative tasks across multiple languages makes it particularly useful for multilingual environments and low-resource language tasks. For the study, mBART50 is highly relevant because it can effectively handle Bengali, one of the 50 languages in its pretraining. Some studies [71] and [72] further highlight the capabilities of multilingual models such as mBART50 in solving math word problems.

4.2.2 Large-Language Models

When using large language models (LLMs), we evaluated their performance through in-context zero-shot and few-shot prompting, emphasizing the Chain of Thought (CoT) approach for reasoning-intensive tasks. This method involved designing prompts that guided the models to provide step-by-step explanations, enhancing clarity and interpretability. For each task, we tested a variety of prompts, adhering to uniform instructions and formats as outlined by the OpenAI Chat playground and Prompt Source (Bach et al., 2022) [73]. Figure 4.2 illustrates the Schematic diagram of our proposed prompting approach.

Additionally, we applied fine-tuning to structure the CoT prompting more effectively. To balance creativity and coherence, we set the temperature to one for all models, enabling them to generate mathematical equations with detailed, logical explanations. This approach fully leveraged the models' potential while ensuring the clarity and precision necessary to solve complex mathematical problems.

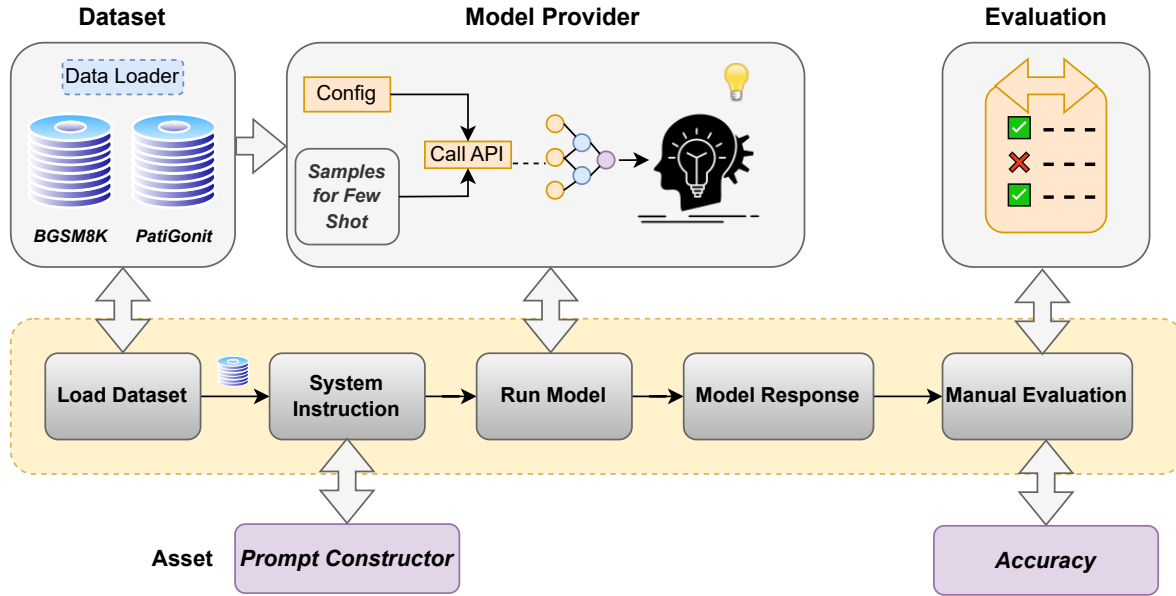


Figure 4.2: Schematic diagram of our proposed prompting approach

Gpt-4o and Gpt-3.5 Turbo

GPT models, developed by OpenAI, are state-of-the-art large language models designed for a wide range of natural language processing tasks. Notable examples include GPT-4o [30] with trillions of parameters and GPT-3.5 Turbo [31] with approximately 175 billion parameters. These models are known for their unparalleled ability to understand and generate human-like text, handle complex instructions, and solve intricate tasks with high precision.

Their success stems from extensive training on diverse datasets and leveraging advanced attention mechanisms to capture contextual nuances. In mathematical problem-solving, GPT models excel by employing the Chain of Thought (CoT) approach, generating detailed intermediate reasoning steps that enhance logical accuracy and interpretability. By breaking problems into smaller, manageable parts, CoT ensures systematic reasoning and correct final results. This capability makes GPT models exceptionally effective for tasks requiring deep comprehension, step-by-step execution, and transparent problem-solving processes.

Llama (Large Language Model Meta AI)

LLaMA (Large Language Model Meta AI) is a series of advanced language models developed by MetaAI. These models, such as LLaMA-3 70B [32], LLaMA-3 8B [32], LLaMA-3.1 70B [32], LLaMA-3.2 90B [32], and LLaMA-3.3 70B [32], represent significant strides in NLP with their efficient scaling and fine-tuning capabilities. Unlike other large models, LLaMA focuses on optimizing performance through parameter-efficient designs, making them particularly effective for specific tasks even with smaller sizes. The LLaMA-3 series

introduces improvements in token processing, context understanding, and task adaptability, demonstrating remarkable performance across a variety of benchmarks. Despite having fewer parameters than GPT-4, LLaMA models excel in tasks like mathematical reasoning by leveraging optimized architectures and structured pre-training to produce accurate, coherent outputs. They effectively apply a structured reasoning approach inspired by the Chain of Thought (CoT) process, enabling step-by-step problem-solving by breaking complex tasks into intermediate steps. This systematic methodology allows LLaMA models to process and solve tasks efficiently while maintaining clarity and precision. Their ability to utilize CoT-style reasoning ensures reliable solutions and highlights their effectiveness in tackling challenges that demand logical progression and detailed analysis.

4.3 Prompting Techniques

While working with large language models (LLMs), we aimed to make our instructions clear and easy to follow so the models could better understand our tasks and give accurate responses. This was important because LLMs often generate different answers depending on how the prompts are designed. Creating these prompts was a step-by-step process with challenges due to the way information is represented differently across models. Our study focused on improving instructions and prompts to guide the models in acting like math instructors and solving math problems with accurate solutions. We also created two sets of instructions and developed unique prompt styles for each of our datasets. Effective prompting is particularly important for encouraging the models to adopt a Chain of Thought (CoT) approach. Chain-of-thought (CoT) prompting has several key advantages for improving how language models reason:

1. Helps models break complex problems into smaller, easier steps, allowing them to focus on tasks that need more reasoning.
2. Gives us a clearer view of how the model arrives at an answer, making it easier to spot where the reasoning went wrong (though fully explaining a model's thinking is still challenging).
3. Can be used for various tasks, like solving math problems, understanding everyday situations, and manipulating symbols, and can potentially apply to any task that humans can solve using language.
4. CoT reasoning can be triggered in large language models by simply including examples of reasoning steps in a few-shot prompt.

4.3.1 Zero Shot Prompting

Zero-shot prompting is particularly useful for testing a model’s inherent capabilities and adaptability without the need for extra fine-tuning or example-based context. In our approach, we use zero-shot prompting by providing chain-of-thought instructions that clearly describe the task and outline the expected output, without offering any example math problems. This method allows the LLMs to create their own context and refine their reasoning process, leading to more accurate results. In Figure 4.3, we showcase an example of a zero-shot prompt, highlighting the instructions and placeholders for both the input and output.

role: system

content: *You are a Bengali Math Word Problem solver who is very skilled at solving high school math word problems in Bengali. You will be given a Bengali [MATH] word problem where you need to first understand the intermediate steps to find the [SOLUTION] in Bengali with proper equations step by step. Your response format should be as follows: [SOLUTION] = intermediate step by step solution of the math word problem in Bengali.*

(a) System Instruction

role: user

content: *রুমি এবং আরও ৪ জন মহিলা প্রতিদিন এক সাথে হাটে। সব মহিলারা একসাথে মোট ৩ মাইল হাঁটেন। তার উপরে, জেমি ৬ দিনের জন্য প্রতিদিন অতিরিক্ত ২ মাইল হাঁটে যখন তার বন্ধু সুমি ৬ দিনে সেই পরিমাণের অর্ধেক হাঁটে। যদি তারা সপ্তাহে ৬ দিন একই পথে হাঁটে, তাহলে মহিলারা মোট কত মাইল হাঁটবে?*

(b) User Question

Figure 4.3: Instruction (Prompt-1) for BGSM8K Dataset (Zero Shot)

4.3.2 Few Shot Prompting

In our approach, we focused on leveraging few-shot learning, which has been shown to outperform zero-shot learning, as demonstrated in studies by Brown et al. (2020) [28] and Ahuja et al. (2023) [74]. We conducted experiments with GPT-4, GPT-3.5, and the LLaMA-3 series, using training data to enhance their performance. To optimize API costs, we manually included five few-shot prompts. For our two datasets, we designed distinct prompts. Several key steps were taken: we taught the model how to approach and reason through the task, followed by question-and-answer prompts for generating answers. Finally, math problem tests were conducted to assess the model’s performance. Our method further augments each few-shot example by incorporating a chain of thought, improving accuracy and clarity in the results. In Figure 4.4 and 4.5, we showcase an examples of few-shot prompt for both of our proposed datasets.

role: system

content: You are a Bengali Math Word Problem solver who is very skilled at solving high school math word problems in Bengali. You will be given a Bengali [MATH] word problem where you need to first understand the intermediate steps to find the [SOLUTION] in Bengali with proper equations step by step. Your response format should be as follows:
[SOLUTION] = intermediate step by step solution of the math word problem in Bengali.

(a) System Instruction

Shot-1

role: user

content: জোহান যখন ১০ বছর তখন তার বাবা তার বয়সের চারগুণ এবং তার মা তার বাবার চেয়ে ২ বছরের ছোট। জোহানের জন্মের সময় তার মায়ের বয়স কত ছিল?

role: assistant

content: জোহান যখন ১০ বছর তখন তার বাবার বয়স $10 \times 8 = 80$ বছর। তার মায়ের বয়স $80 - 2 = 78$ বছর। যখন জোহান জন্মগ্রহণ করেন, তখন তার মায়ের বয়স $78 - 10 = 68$ বছর। ##### ৬৮

Shot-2 . . .

⋮

Shot-5 . . .

(b) 5 Shot Prompting

role: user

content: রুমি এবং আরও ৪ জন মহিলা প্রতিদিন এক সাথে হাটে। সব মহিলারা একসাথে মোট ৩ মাইল হাঁটেন। তার উপরে, জেমি ৬ দিনের জন্য প্রতিদিন অতিরিক্ত ২ মাইল হাঁটে যখন তার বন্ধু সুমি ৬ দিনে সেই পরিমাণের অর্ধেক হাঁটে। যদি তারা সপ্তাহে ৬ দিন একই পথে হাঁটে, তাহলে মহিলারা মোট কত মাইল হাঁটবে?

(c) User Question

Figure 4.4: Instruction (Prompt-1) for BGSM8K Dataset (Few Shot)

role: system

content: You are an expert Bengali Math teacher who is very skilled at solving high school math word problems in Bengali. You will be given a Bengali MATH where you need to find just the EQUATION in Bengali

(a) System Instruction

Shot-1

role: user

content: সুমন একটি দোকান থেকে ৫টি পেন্সিল কিনেছে। প্রতিটি পেন্সিলের দাম ৮ টাকা। সুমন সব পেন্সিলের জন্য কত টাকা খরচ করেছে?

role: assistant

content: $k = 5 \times 8 = 80$

Shot-2 . . .

⋮

Shot-5 . . .

(b) 5 Shot Prompting

role: user

content: একটি বিন্ডিংয়ে ১২টি কক্ষ আঁকার জন্য একজন চিত্রশিল্পীর প্রয়োজন হয়। প্রতিটি রুম রং করতে ৭ ঘন্টা সময় লাগে। যদি সে ইতিমধ্যেই ৫টি ঘর রং করে ফেলে, তাহলে বাকিটা রং করতে তার আর কত সময় লাগবে?

(c) User Question

Figure 4.5: Instruction (Prompt-1) for PatiGonit Dataset (Few Shot)

4.4 Fine Tuning

4.4.1 Gpt-3.5 Standard Tuning

Fine-tuning (OpenAI, 2023a) [75] enhances few-shot learning by training on a larger set of examples, resulting in improved performance across various tasks. In this approach, we integrated chain of thought to encourage models to generate step-by-step reasoning in their solutions. Once a model is fine-tuned, fewer examples are required in the prompt, which reduces costs and enables faster response times. The fine-tuning process consists of the following steps, which differ from few-shot prompting:

1. Compile and upload the required data for training, ensuring the inclusion of chain-of-thought examples where necessary
2. Develop a new model that is optimized for better performance, focusing on improving reasoning abilities with intermediate steps
3. Analyze the outcomes and make necessary adjustments to ensure that the model produces clear, logical reasoning along with its final answers
4. Utilize the refined and optimized model, leveraging the chain of thought to maintain accurate, interpretable problem-solving

We chose to fine-tune the recommended model, GPT-3.5-turbo-0125. During this process, we varied only the number of epochs to optimize the model's performance. To build the fine-tuned model, we used 50 examples with appropriate instructions, including explicit chain-of-thought reasoning for complex problems. This ensured the model not only provided answers but also demonstrated the thought process behind solving math problems. The examples followed the JSONL file format as outlined in the OpenAI API documentation. Figure 4.6 shows an example of how the JSON file was structured to create the fine-tuned model.

Our BGSM8K dataset required proper chain-of-thought reasoning to produce correct solutions, so fine-tuning was essential. The models were trained to break down complex problems into intermediate steps, ensuring that the reasoning behind the answers was explicit and interpretable. In contrast, the BMAWPS dataset only contains math word problem equations, which require less complex reasoning. For this dataset, GPT-3.5 Turbo and GPT-4 models performed well with few-shot prompting, so extensive fine-tuning was not necessary.

1. **"messages": [{"role": "system",
"content": "You are a Bengali Math Word Problem solver who is very skilled at solving high school math word problems in Bengali. You will be given a Bengali [MATH] word problem where you need to first understand the intermediate steps to find the [SOLUTION] in Bengali with proper equations step by step. Your response format should be as follows: [SOLUTION] = intermediate step by step solution of the math word problem in Bengali."}, {"role": "user",
"content": "রুমি এবং আরও ৪ জন মহিলা প্রতিদিন এক সাথে হাটে। সব মহিলারা একসাথে মোট ৩ মাইল হাঁটেন। তার উপরে, জেমি ৬ দিনের জন্য প্রতিদিন অতিরিক্ত ২ মাইল হাঁটে যখন তার বন্ধু সুমি ৬ দিনে সেই পরিমাণের অর্ধেক হাঁটে। যদি তারা সপ্তাহে ৬ দিন একই পথে হাঁটে, তাহলে মহিলারা মোট কত মাইল হাঁটবে?"}, {"role": "assistant", "content": "দলে মোট ৫ জন মহিলা রয়েছেন। তারা সকলেই ৬ দিন প্রতিদিন ৩ মাইল হাঁটেন তাই $৩ \times ৬ = ১৮$ মাইল। রুমি ৬ দিনের প্রতিদিন অতিরিক্ত ২ মাইল হাঁটে তাই $২ \times ৬ = ১২$ মাইল। সুমি রুমির অর্ধেক দূরত্ব অতিক্রম করে তাই $১২/২ = ৬$ মাইল। একসাথে, মহিলারা $১৮ + ১২ + ৬ = ৩৬$ মাইল হাঁটেন #### ৩৬"}]**
- 2.
- .
- .
- 50.

Figure 4.6: Structure of JSON file format

4.4.2 LoRA Fine Tuning

For our BGSM8K dataset, we employed Low-Rank Adaptation (LoRA) (Hu et al., 2021) [29] for fine-tuning. LoRA improves fine-tuning by learning two smaller weight-updating matrices, allowing the large model to remain frozen while only the smaller matrices are updated with new data. After training, the updated matrices are recombined with the original weights to create the final model. This method significantly reduces the number of parameters and accelerates the fine-tuning process while saving storage. As part of this process, we also integrated chain-of-thought examples into the fine-tuning dataset to encourage the model to generate logical, step-by-step reasoning. Figure 4.7 shows an overview of the process.

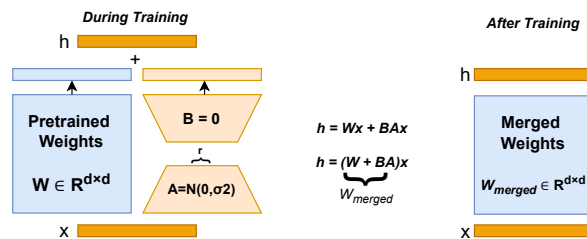


Figure 4.7: LoRA Fine-Tuning Process: During and After Training (LoRA, 2024) [1]

In summary, this section outlines the proposed approaches for solving Bengali math word problems using models like BanglaT5, mBART-50, GPT-4, GPT-3.5 Turbo, and LLaMA. It highlights techniques like zero-shot and few-shot prompting, emphasizing the Chain of Thought (CoT) approach for step-by-step reasoning. Fine-tuning methods, including standard fine-tuning and Low-Rank Adaptation (LoRA), are discussed to enhance model performance on datasets like BGSM8K. These approaches aim to improve accuracy, logical reasoning, and the interpretability of solutions.

Chapter 5

Experimental Setup

The experimental setup outlines the computational environment, data preparation, evaluation metrics, and hyperparameter tuning strategies employed in this study. We conducted experiments on Google Colab with a Tesla T4 GPU and implemented models using Python and PyTorch. The datasets were divided into appropriate train-test-validation splits to ensure robust training and evaluation. Multiple evaluation metrics, including BLEU scores and solution accuracy, were used to assess model performance, with a focus on mathematical correctness. Hyperparameter tuning was conducted systematically to optimize performance while accounting for resource constraints.

5.1 Baseline Setup

Our work was conducted on Google Colab Notebook with Python 3.10.12, PyTorch 2.0.1, a Tesla T4 GPU (15 GB), 12.5 GB of RAM, and 64 GB of disk space.

5.2 Train-Test-Validation splits

For our BGSM8k dataset’s prompting and evaluation, we divided the dataset into two parts: 80% for testing and 20% for training. Specifically, 7,033 examples were assigned for training, while 1,759 examples were reserved for testing. Prompting and finetuning samples were taken from training sets and evaluation was done by testing samples.

We split the PatiGonit dataset into three parts: 80% for training, 10% for validation, and 10% for testing. This 80:10:10 split offers a well-balanced approach, providing ample data for the model to learn effectively while retaining a sufficient validation set for tuning hyperparameters and a robust test set for evaluating performance on unseen data.

5.3 Evaluation metrics

5.3.1 BGSM8K Evaluation

Chapter 1 problem statement states that, the model's output can be represented as:

$$f(x) = (s_1, s_2, \dots, s_k, r_{pred})$$

where s_1, s_2, \dots, s_k are the intermediate steps and r_{pred} is the predicted final result.

For evaluation, the intermediate reasoning steps s_1, s_2, \dots, s_k are not considered. Only the final result r_{pred} is compared with the ground truth r_{true} :

$$\text{Evaluation}(f(x), r_{true}) = \begin{cases} 1 & \text{if } r_{pred} = r_{true}, \\ 0 & \text{if } r_{pred} \neq r_{true}. \end{cases} \quad (5.1)$$

The model is considered to have successfully solved the problem if: $r_{pred} = r_{true}$

If the dataset contains N problems x_1, x_2, \dots, x_N , the accuracy A is computed as:

$$A = \frac{1}{N} \sum_{i=1}^N \mathbf{1}(r_{pred,i} = r_{true,i})$$

where $\mathbf{1}(\cdot)$ is the indicator function, which equals 1 if the condition is true and 0 otherwise.

5.3.2 PatiGonit Evaluation

<i>Actual Equation</i>	<i>Predicted Equation</i>	<i>Bleu Score</i>
1) ৩৯*১৭=৬৬৩	1) ৩৯/১৭=৬৬৩	0.3218
2) (৭.০*(১২.০-৫.০)) = ৪৯	2) ((৫.০-১২.০)*৭.০) = ৪৯	0.1524

Figure 5.1: Bleu Score examples

The first metric used to evaluate the model's performance is the BLEU (Bilingual Evaluation Understudy) score, in Figure 5.1 case from our data, the reference equation 2 and the predicted equation 2 had a low BLEU score of 0.1524 despite being mathematically correct. On the other hand the reference equation 1 and the predicted equation 1 had a BLEU score

of 0.3218, which is higher than the equation 1, even though it's wrong. This happens because BLEU measures surface-level token similarity rather than evaluating the mathematical equivalence of the equations [17]. Hence, minor differences in the order of terms can lead to lower BLEU scores even when the solution is accurate.

To address this, we used solution accuracy, which compares the final solution from the predicted equation with the correct answer. This metric more effectively measures the model's ability to solve math word problems by focusing on the correctness of the numerical result, regardless of minor variations in equation structure.

Chapter 1 problem statement states that, The model's output can be represented as:

$$f(x) = (e, r_{\text{pred}})$$

where e is the equation generated by the model, and r_{pred} is the predicted final result.

Evaluation focuses on comparing r_{pred} with the ground truth r_{true} , while optionally verifying the correctness of the generated equation e .

5.4 Hyperparameter Tuning

Table 5.1: Hyper Parameters

Hyperparameter	Enforced Variations
Dropout	0.1
Batch Size	8, 16
Epoch	5, 10, 15
Learning Rate	1e-4

Table 5.1 presents the hyperparameters we applied for our Transformer-based approach across all models during training. The selection of hyperparameters was based on standard practices and preliminary experimentation to optimize the performance of the model while considering resource constraints. Dropout was fixed at 0.1 to prevent overfitting, as it is commonly effective in transformer models. For batch size, we tested 8 and 16 to balance memory usage and training efficiency, while the number of epochs was varied (5, 10, 15) to capture the effect of extended training. The learning rate was set at 1e-4, a standard value for stable convergence in transformer-based architectures. For mBART50, however, we were constrained to a batch size of 8 with 5 epochs due to resource limitations in Google Colab. Despite this limitation, we maintained consistency in other hyperparameters across all models to ensure a fair comparison.

In summary, The experiments were conducted on Google Colab using a Tesla T4 GPU, with datasets split into training, validation, and test sets for BGSM8K and PatiGonit. Evaluation

metrics included BLEU scores and solution accuracy, with a focus on mathematical correctness. Hyperparameter tuning was performed on dropout, batch size, epochs, and learning rate, with variations to optimize performance under resource constraints.

Chapter 6

Result Analysis

The result analysis section provides a detailed evaluation of the model's performance on the BGSM8k and PatiGonit datasets. We examine accuracy, BLEU scores, and solution correctness to assess the effectiveness of our proposed approaches.

6.1 Transformer based results

Table 6.1: Performance of the proposed models on "PatiGonit" dataset

Model Name	Batch Size	Epoch	Bleu	Accuracy
Transformer	8	5	68.45	47.50%
		10	82.89	72.05%
		15	87.95	77.30%
	16	5	80.75	66.00%
		10	85.28	74.05%
		15	82.18	72.35%
mT5	8	5	86.41	85.70%
		10	93.61	96.20%
		15	94.68	97.30%
	16	5	40.16	14.40%
		10	93.07	95.80%
		15	94.19	96.80%
BanglaT5	8	5	89.14	92.40%
		10	94.05	95.80%
		15	94.06	95.50%
	16	5	93.31	95.80%
		10	32.50	1.30%
		15	89.96	92.20%
mBART50	8	5	95.74	97.20%

The performance analysis of the models as shown in Table 6.1 indicates that mBART50 achieved the highest accuracy of 97.20% with a BLEU score of 95.74, using a batch size of 8 and 5 epochs. This demonstrates mBART50's strong capability in handling Bengali math word problems. MT5 also performed exceptionally well, reaching a maximum accuracy of 97.30% and a BLEU score of 94.68 with a batch size of 8 and 15 epochs. BanglaT5 showed competitive performance, with its best accuracy at 95.80% and a BLEU score of 94.06 using a batch size of 8 and 15 epochs. However, when the batch size was increased to 16, BanglaT5's accuracy drastically dropped, indicating sensitivity to hyperparameter settings. The Transformer model, while showing gradual improvement with more epochs and larger batch sizes, reached a peak accuracy of 77.30%, which is significantly lower than that of the other transformer-based models. This result highlights the superior performance of mBART50, MT5, and BanglaT5 over the basic Transformer model in accurately solving Bengali math word problems. mT5 achieved the highest accuracy among all the models, with a peak performance of 97.30% accuracy, and a BLEU score of 94.68 when using a batch size of 8 and 15 epochs closely followed by mBART50 which has the accuracy of 97.20%. This demonstrates mT5's superior capability in translating Bengali math word problems into accurate equations. Its performance remained consistently high across different configurations, indicating its robustness and effectiveness in handling the linguistic complexity of Bengali text. These results show that mT5 is exceptionally well suited for this task, outperforming other models in terms of overall accuracy.

6.2 Large language Models (LLMs) based results

The results in Table 6.2 show key distinctions between Standard prompting and Chain of Thought (CoT) prompting, highlighting their respective strengths and weaknesses.

Table 6.2: Chain of Thoughts prompting outperforms Standard prompting for various Large language models for the "BGSM8K" dataset. A benchmark has been set for the "PatiGonit" dataset using Few Shot prompting technique. All metrics are Accuracy (%). (*) represents that the accuracy is correct but the responses of the model was in English.

Models	Parameters	BGSM8K				PatiGonit
		Standard		Chain of Thoughts (CoT)		CoT
		Zero Shot	Few (5) Shot	Zero Shot	Few (5) Shot	Few (5) Shot
GPT-3.5	175B	23.2	24.2	30.6 _(+7.4)	24.8 _(+0.6)	86.4
GPT-4o	1.3T(approx.)	79.2	80.8	80.4 _(+1.2)	83.8 _(+3.0)	99.0
Llama-3	70B	62.8*	65.3	70.4 _(+7.6)	67.6 _(+2.3)	96.0
	8B	24.8*	19.2	47.6 _(+22.8)	33.6 _(+14.4)	94.8
Llama-3.1	70B	78.0	70.0	79.5 _(+1.5)	70.2 _(+0.2)	96.2
Llama-3.2	90B	75.0	69.1	79.0 _(+4.0)	72.8 _(+3.7)	96.5
Llama-3.3	70B	83.2	80.6	87.8 _(+4.6)	81.4 _(+0.8)	96.9

In Standard prompting, GPT-3.5, with 175 billion parameters, achieves relatively modest performance, scoring 23.2% in the zero-shot setting and 24.2% in the few-shot setting. In contrast, GPT-4o, with approximately 1.3 trillion parameters, demonstrates a substantial improvement, achieving 79.2% and 80.8% in zero-shot and few-shot, respectively, showing the benefits of increased parameter size. Among Llama models, Llama-3 (70B) achieves 62.8% (zero-shot) and 65.3% (few-shot), while its smaller variant, Llama-3 (8B), performs significantly lower, with 24.8% (zero-shot) and 19.2% (few-shot). Newer versions, such as Llama-3.3 (70B), outperform the earlier models, achieving 83.2% in zero-shot and 80.6% in few-shot settings, marking a notable improvement. This comparison highlights the role of parameter scaling and iterative advancements in improving task performance, with GPT-4o and Llama-3.3 emerging as the top-performing models.

In Chain of Thought (CoT) prompting, CoT prompting significantly enhances model performance, especially in Zero-Shot settings, as it allows the models to break down complex problems into smaller, logical steps. GPT-4o leads the way, achieving 80.4% Zero-Shot and 83.8% Few-Shot accuracy on the BGSM8K dataset, demonstrating its ability to leverage CoT to handle complex reasoning tasks effectively. Among the LLaMA models, LLaMA-3.3 (70B) achieves the highest Zero-Shot accuracy at 87.8%, outperforming both the larger LLaMA-3.2 (90B) and the smaller variants. The improvement is particularly pronounced for smaller models, such as LLaMA-3 (8B), where Zero-Shot CoT accuracy jumps from 24.8% (Standard) to 47.6%, showcasing how CoT helps smaller models perform better in tasks requiring step-by-step reasoning.

In the PatiGonit dataset, CoT Few-Shot achieves exceptional results, with GPT-4o nearing perfection at 99% and LLaMA-3.3 (70B) following closely with 96.9% which are better than the transformer models. These results highlight CoT's ability to maximize the reasoning potential of models, regardless of the dataset's complexity.

In most cases, LLMs perform better in zero-shot settings compared to few-shot (5) prompting. This is because, in zero-shot, models rely entirely on their pre-trained knowledge to handle complex reasoning. In few-shot settings, the limited and often intricate examples may fail to capture broader mathematical patterns, making it harder for the models to generalize. As a result, zero-shot sometimes outperforms few-shot when the provided examples are insufficiently representative.

We proposed the “PatiGonit” and “BGSM8K” dataset, which have not been the subject of any prior research so far.

In summary, in Standard prompting, GPT-4o leads with 79.2% (zero-shot) and 80.8% (few-shot), while LLaMA-3.3 (70B) follows at 83.2% (zero-shot). In CoT prompting, models excel in reasoning tasks, with GPT-4o achieving 80.4% (zero-shot) and 83.8% (few-shot) and LLaMA-3.3 (70B) reaching 87.8% (zero-shot). Smaller models, like LLaMA-3 (8B),

improve drastically with CoT. On the PatiGonit dataset, GPT-4o hits 99% and LLaMA-3.3 achieves 96.9% which are better than transformer models.

Zero-shot often outperforms few-shot because it relies entirely on the model’s pre-trained knowledge, which is broad and generalized. In contrast, few-shot depends on limited examples that may not adequately represent the problem, making it harder for models to generalize effectively sometimes. Some Correct and Incorrect examples of model’s responses along with the prompting techniques are given in Appendix A.

6.3 Fine Tuning based results

6.3.1 LoRA Fintune based results

Table 6.3: Ablation Study with three variations of LoRA finetuning on GSM8K dataset. All metrics are Accuracy (%).

Parameter	Baseline	Higher Rank with Dropout	Memory Efficient with Lower Batch Size
LoRA Parameters			
r	16	32	16
lora alpha	16	32	16
lora dropout	0	0.1	0.1
target modules	{q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj}	{q_proj, k_proj, v_proj, o_proj}	{q_proj, v_proj, o_proj}
use gradient checkpointing	unsloth	True	True
Training Arguments			
learning rate	2e-4	1e-4	2e-4
per device train batch size	2	4	1
gradient accumulation steps	4	2	8
warmup steps	5	10	5
max steps	60	100	60
Performance	13.0	12.0	17.0

From Table 6.3,

Baseline Configuration (Finetune 1): This setup, characterized by a rank of 16, no dropout, and a moderate learning rate of 2e-4, achieved an accuracy of 13%. The simplicity of this configuration likely facilitated quicker optimization but also increased the risk of overfitting due to the lack of regularization. The moderate parameters provided a reference point

for evaluating more complex setups, showing the limitations of basic fine-tuning without dropout.

Higher Rank with Dropout (Finetune 2): Incorporating a higher rank (32) and dropout (0.1), this configuration aimed to increase model capacity and regularization. However, it achieved a slightly lower accuracy of 12%, despite its cautious learning rate of $1e-4$. This result suggests that the added complexity and regularization may not align well with the dataset's requirements, possibly due to over-parameterization or insufficient data to fully utilize the additional capacity.

Memory-Efficient with Lower Batch Size (Finetune 3): The third configuration utilized memory-efficient adjustments, including a lower batch size and increased gradient accumulation steps, alongside a higher learning rate of $2e-4$. This setup achieved the highest accuracy of 17%. The reduced batch size likely allowed for better gradient updates, while the increased accumulation steps improved learning stability. The higher learning rate might have further facilitated effective convergence by optimizing balancing and learning dynamics.

6.3.2 Gpt-3.5 Finetune result

Table 6.4: Performance comparison of finetuned Gpt-3.5 with Gpt-4o. All metrics are Accuracy (%).

Models	Learning Rate	Epochs	Performance
Gpt-3.5 turbo-0125	0.1	5	12.6
		10	23.4
		15	12.6
Gpt-4o (Standard Prompting Zero Shot)	-	-	79.2

Table 6.4 shows the performance comparison between fine-tuned GPT-3.5 (turbo-0125) and GPT-4.0 (using standard zero-shot prompting), reveals significant differences in accuracy. Fine-tuning GPT-3.5 with a learning rate of 0.1 over 10 epochs achieved the best performance among the fine-tuned configurations at 23.4%. However, increasing the epochs to 15 did not yield any further improvements, with the performance plateauing at 12.6%, matching the result from 5 epochs. This suggests that fine-tuning at higher epochs might lead to overfitting or diminished returns.

In contrast, GPT-4.0, without any fine-tuning, demonstrated a far superior performance of 79.2% using its standard zero-shot prompting capability. This highlights GPT-4.0's inherent ability to generalize and perform well on tasks without the need for task-specific training, significantly outperforming GPT-3.5 even with fine-tuning.

In summary, the results show that mBART50, MT5, and BanglaT5 outperformed the Transformer model, with mT5 achieving the highest accuracy of 97.30% on the BGSM8K dataset. In large language models (LLMs), GPT-4o and Llama-3.3 achieved significant performance improvements using Chain of Thought (CoT) prompting, with GPT-4o reaching 83.8% accuracy on BGSM8K. Fine-tuning with LoRA showed varying results, with the memory-efficient configuration achieving the highest accuracy of 17%. GPT-3.5's fine-tuning was less effective, with its best performance at 23.4%, while GPT-4o performed significantly better in zero-shot settings at 79.2%.

Chapter 7

Conclusion and Future Work

In this study, we have successfully implemented the transformer based models and enhanced the ability of large language models (LLMs) to solve Bengali Math Word Problems (MWPs). We established BGSM8K and PatiGonit as robust training and evaluation datasets. Transformer based models were evaluated using the PatiGonit dataset, where mT5 gave us the highest accuracy of 97.30%. Few-shot learning and fine-tuning worked well in our experiments, with GPT-4o outperforming GPT-3.5 Turbo-16k in few-shot scenarios but struggling with more complex tasks. A significant contribution of our work is the introduction of Low-Rank Adaptation (LoRA) fine-tuning techniques specifically tailored for our BGSM8K dataset. Our work not only advances the field of Bengali natural language processing but also contributes valuable resources and methodologies for future research in educational AI tools.

Future work will refine fine-tuning techniques, expand datasets, explore various prompting strategies, and implement Retrieval-Augmented Generation (RAG) to counter LLM hallucinations. We also aim to apply these methods to other low-resource languages.

Chapter 8

Limitations

Our study faced several limitations that impacted its scope and effectiveness. Translating English math word problems into Bengali proved challenging due to cultural differences, including names and references, leading to ambiguity and inaccuracies. Additionally, the PatiGonit dataset was dominated by basic arithmetic problems, limiting the models' ability to address advanced queries involving complex equations and multiple operators. This lack of diversity, compounded by the absence of high-quality Bengali datasets and dialectal variations, further constrained the models' potential. Despite GPT-4o's strong performance, the process required substantial computational power, and fine-tuning techniques like LoRA demand further optimization. Finally, our findings remain preliminary, requiring additional testing across diverse datasets and models to validate the results.

References

- [1] LoRA, “Lora,” 2024. Accessed: 2024-06-14.
- [2] G. Yigit and M. F. Amasyali, “Enhancing multiple-choice question answering through sequential fine-tuning and curriculum learning strategies,” *Knowledge and Information Systems*, vol. 65, no. 11, pp. 5025–5042, 2023.
- [3] L. Wu, P. Wu, and X. Zhang, “A seq2seq-based approach to question answering over knowledge bases,” in *Semantic Technology: 9th Joint International Conference, JIST 2019, Hangzhou, China, November 25–27, 2019, Revised Selected Papers 9*, pp. 170–181, Springer, 2020.
- [4] A. Rogers, M. Gardner, and I. Augenstein, “Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension,” *ACM Computing Surveys*, vol. 55, pp. 1 – 45, 2021.
- [5] G. Yigit and M. F. Amasyali, “Ask me: A question answering system via dynamic memory networks,” *2019 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 1–5, 2019.
- [6] Z. Xie and S. Sun, “A goal-driven tree-structured neural model for math word problems,” in *International Joint Conference on Artificial Intelligence*, 2019.
- [7] J. Zhang, R. K.-W. Lee, E.-P. Lim, W. Qin, L. Wang, J. Shao, and Q. Sun, “Teacher-student networks with multiple decoders for solving math word problem,” in *International Joint Conference on Artificial Intelligence*, 2020.
- [8] Z. Liang, J. Zhang, L. Wang, W. Qin, Y. Lan, J. Shao, and X. Zhang, “Mwp-bert: Numeracy-augmented pre-training for math word problem solving,” in *NAACL-HLT*, 2021.
- [9] 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,” in *AAAI Conference on Artificial Intelligence*, 2019.

- [10] J. Zhang, L. Wang, R. K.-W. Lee, Y. Bin, Y. Wang, J. Shao, and E.-P. Lim, “Graph-to-tree learning for solving math word problems,” in *Annual Meeting of the Association for Computational Linguistics*, 2020.
- [11] Y. Shen and C. Jin, “Solving math word problems with multi-encoders and multi-decoders,” in *International Conference on Computational Linguistics*, 2020.
- [12] E. A. Feigenbaum, “Some challenges and grand challenges for computational intelligence,” *Journal of the ACM (JACM)*, vol. 50, no. 1, pp. 32–40, 2003.
- [13] A. Vaswani, “Attention is all you need,” *Advances in Neural Information Processing Systems*, 2017.
- [14] L. Xue, “mt5: A massively multilingual pre-trained text-to-text transformer,” *arXiv preprint arXiv:2010.11934*, 2020.
- [15] A. Bhattacharjee, T. Hasan, W. U. Ahmad, and R. Shahriyar, “Banglanlg and banglat5: Benchmarks and resources for evaluating low-resource natural language generation in bangla,” *arXiv preprint arXiv:2205.11081*, 2022.
- [16] Y. Tang, C. Tran, X. Li, P.-J. Chen, N. Goyal, V. Chaudhary, J. Gu, and A. Fan, “Multilingual translation with extensible multilingual pretraining and finetuning,” *arXiv preprint arXiv:2008.00401*, 2020.
- [17] A. K. Nayak, R. Patwari, and V. Subramanian, “Mathbot—a deep learning based elementary school math word problem solver,” <https://www.semanticscholar.org/paper/MathBot-%E2>, vol. 80, p. 7.
- [18] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, *et al.*, “Training verifiers to solve math word problems,” *arXiv preprint arXiv:2110.14168*, 2021.
- [19] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, *et al.*, “Scaling language models: Methods, analysis & insights from training gopher,” *arXiv preprint arXiv:2112.11446*, 2021.
- [20] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” *Advances in neural information processing systems*, vol. 35, pp. 24824–24837, 2022.
- [21] . Wikipedia, “List of languages by total number of speakers.” Accessed: 2024-11-22.
- [22] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom, “Program induction by rationale generation: Learning to solve and explain algebraic word problems,” *arXiv preprint arXiv:1705.04146*, 2017.

- [23] A. Patel, S. Bhattamishra, and N. Goyal, “Are nlp models really able to solve simple math word problems?,” *arXiv preprint arXiv:2103.07191*, 2021.
- [24] S. Roy and D. Roth, “Solving general arithmetic word problems,” *arXiv preprint arXiv:1608.01413*, 2016.
- [25] T.-R. Chiang and Y.-N. Chen, “Semantically-aligned equation generation for solving and reasoning math word problems,” *arXiv preprint arXiv:1811.00720*, 2018.
- [26] A. Amini, S. Gabriel, P. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi, “Mathqa: Towards interpretable math word problem solving with operation-based formalisms,” *arXiv preprint arXiv:1905.13319*, 2019.
- [27] X. Chen, C. Liang, A. W. Yu, D. Zhou, D. Song, and Q. V. Le, “Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension,” in *International Conference on Learning Representations*, 2019.
- [28] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [29] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.
- [30] OpenAI, “Gpt 4o,” 2024. Accessed: 2024-06-14.
- [31] OpenAI, “Gpt-3.5k,” 2023. Accessed: 2024-06-14.
- [32] MetaAI, “Llama: Open and efficient foundation language models,” 2024. Accessed: 2024-12-09.
- [33] Y. Wang, X. Liu, and S. Shi, “Deep neural solver for math word problems,” in *Proceedings of the 2017 conference on empirical methods in natural language processing*, pp. 845–854, 2017.
- [34] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, “Translating a math word problem to an expression tree,” *arXiv preprint arXiv:1811.05632*, 2018.
- [35] B. Zhang, D. Xiong, J. Su, H. Duan, and M. Zhang, “Variational neural machine translation,” *arXiv preprint arXiv:1605.07869*, 2016.
- [36] D. Huang, J. Liu, C.-Y. Lin, and J. Yin, “Neural math word problem solver with reinforcement learning,” in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 213–223, 2018.

- [37] J. Zhang, L. Wang, R. K.-W. Lee, Y. Bin, Y. Wang, J. Shao, and E.-P. Lim, “Graph-to-tree learning for solving math word problems,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3928–3937, 2020.
- [38] S. Li, L. Wu, S. Feng, F. Xu, F. Xu, and S. Zhong, “Graph-to-tree neural networks for learning structured input-output translation with applications to semantic parsing and math word problem,” *arXiv preprint arXiv:2004.13781*, 2020.
- [39] Y. Shen and C. Jin, “Solving math word problems with multi-encoders and multi-decoders,” in *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 2924–2934, 2020.
- [40] Z. Liang, J. Zhang, L. Wang, W. Qin, Y. Lan, J. Shao, and X. Zhang, “Mwp-bert: Numeracy-augmented pre-training for math word problem solving,” *arXiv preprint arXiv:2107.13435*, 2021.
- [41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [42] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.
- [43] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [44] P. Clark, O. Tafjord, and K. Richardson, “Transformers as soft reasoners over language,” *arXiv preprint arXiv:2002.05867*, 2020.
- [45] M. Saeed, N. Ahmadi, P. Nakov, and P. Papotti, “Rulebert: Teaching soft rules to pre-trained language models,” *arXiv preprint arXiv:2109.13006*, 2021.
- [46] Z. Liang, S. Bethard, and M. Surdeanu, “Explainable multi-hop verbal reasoning through internal monologue,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1225–1250, 2021.
- [47] Z. Liang, J. Zhang, L. Wang, W. Qin, Y. Lan, J. Shao, and X. Zhang, “Mwp-bert: Numeracy-augmented pre-training for math word problem solving,” *arXiv preprint arXiv:2107.13435*, 2021.
- [48] S. R. Raiyan, M. N. Faiyaz, S. M. J. Kabir, M. Kabir, H. Mahmud, and M. K. Hasan, “Math word problem solving by generating linguistic variants of problem statements,” *arXiv preprint arXiv:2306.13899*, 2023.

- [49] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, “Translating a math word problem to an expression tree,” *arXiv preprint arXiv:1811.05632*, 2018.
- [50] Z. Xie and S. Sun, “A goal-driven tree-structured neural model for math word problems,” in *Ijcai*, pp. 5299–5305, 2019.
- [51] O. Chatterjee, I. Pandey, A. Waikar, V. Kumar, and G. Ramakrishnan, “Warm: A weakly (+ semi) supervised model for solving math word problems,” *arXiv preprint arXiv:2104.06722*, 2021.
- [52] H. Chen, J. He, K. Narasimhan, and D. Chen, “Can rationalization improve robustness?,” *arXiv preprint arXiv:2204.11790*, 2022.
- [53] B. Hancock, M. Bringmann, P. Varma, P. Liang, S. Wang, and C. Ré, “Training classifiers with natural language explanations,” in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, vol. 2018, p. 1884, NIH Public Access, 2018.
- [54] D. Dua, S. Singh, and M. Gardner, “Benefits of intermediate annotations in reading comprehension,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5627–5634, 2020.
- [55] O. Zaidan, J. Eisner, and C. Piatko, “Using “annotator rationales” to improve machine learning for text categorization,” in *Human language technologies 2007: The conference of the North American chapter of the association for computational linguistics; proceedings of the main conference*, pp. 260–267, 2007.
- [56] H. Yao, Y. Chen, Q. Ye, X. Jin, and X. Ren, “Refining language models with compositional explanations,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 8954–8967, 2021.
- [57] P. Hase and M. Bansal, “When can models learn from explanations? a formal framework for understanding the roles of explanation data,” *arXiv preprint arXiv:2102.02201*, 2021.
- [58] Y. Gu, B. D. Mishra, and P. Clark, “Dream: Uncovering mental models behind language models,” *NAACL*, 2022.
- [59] J. Andreas, D. Klein, and S. Levine, “Learning with latent language,” *arXiv preprint arXiv:1711.00482*, 2017.
- [60] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom, “e-snli: Natural language inference with natural language explanations,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.

- [61] N. F. Rajani, B. McCann, C. Xiong, and R. Socher, “Explain yourself! leveraging language models for commonsense reasoning,” *arXiv preprint arXiv:1906.02361*, 2019.
- [62] E. Zelikman, Y. Wu, J. Mu, and N. Goodman, “Star: Bootstrapping reasoning with reasoning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 15476–15488, 2022.
- [63] B. Lester, R. Al-Rfou, and N. Constant, “The power of scale for parameter-efficient prompt tuning,” *arXiv preprint arXiv:2104.08691*, 2021.
- [64] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, “Finetuned language models are zero-shot learners,” *arXiv preprint arXiv:2109.01652*, 2021.
- [65] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja, *et al.*, “Multitask prompted training enables zero-shot task generalization,” *arXiv preprint arXiv:2110.08207*, 2021.
- [66] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, “Training language models to follow instructions with human feedback,” *Advances in neural information processing systems*, vol. 35, pp. 27730–27744, 2022.
- [67] 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,” *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.
- [68] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap, *et al.*, “Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks,” *arXiv preprint arXiv:2204.07705*, 2022.
- [69] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J.-G. Lou, and W. Chen, “On the advance of making language models better reasoners, 2022b,” URL <https://arxiv.org/abs/2206.02336>.
- [70] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, “Self-consistency improves chain of thought reasoning in language models,” *arXiv preprint arXiv:2203.11171*, 2022.
- [71] K. Niyarepola, D. Athapaththu, S. Ekanayake, and S. Ranathunga, “Math word problem generation with multilingual language models,” in *Proceedings of the 15th International Conference on Natural Language Generation*, pp. 144–155, 2022.

- [72] K. Niyarepola, D. Athapaththu, S. Ekanayake, and S. Ranathunga, “Math word problem generation with multilingual language models,” in *Proceedings of the 15th International Conference on Natural Language Generation*, pp. 144–155, 2022.
- [73] S. H. Bach, V. Sanh, Z.-X. Yong, A. Webson, C. Raffel, N. V. Nayak, A. Sharma, T. Kim, M. S. Bari, T. Fevry, *et al.*, “Promptsources: An integrated development environment and repository for natural language prompts,” *arXiv preprint arXiv:2202.01279*, 2022.
- [74] K. Ahuja, H. Diddee, R. Hada, M. Ochieng, K. Ramesh, P. Jain, A. Nambi, T. Ganu, S. Segal, M. Axmed, *et al.*, “Mega: Multilingual evaluation of generative ai,” *arXiv preprint arXiv:2303.12528*, 2023.
- [75] OpenAI, “Gpt 3.5 finetuning,” 2023. Accessed: 2024-06-14.

Appendix A

Prompting Output

Era

Name

Era

asst_qpDGXGSXL7S8cSr4ouMyRBUI

Instructions

You are an expert Bengali Math teacher who is very skilled in solving high school math word problems in Bengali. You will be given a Bengali [MATH] where you need to understand the intermediate steps to find the [SOLUTION] in Bengali with

Model

gpt-4

TOOLS

Functions

Code interpreter

Retrieval

Updated 3/25, 6:55 PM

THREAD thread_jYXOLADxWALdsksfDoXBcTo

1279 tokens

Logs

Era

User

Carefully Analyze the following Bangla word problem, try to understand problems' intermediate steps toward accurate result and provide step by step solution in Bengali of the [MATH] = অপর্ণা বাচ্চা দেখাশোনা করার জন্য ঘন্টায় ১২ টাকা আয় করে। গতকাল সে মাত্র ৫০ মিনিট বাচ্চা দেখাশোনা করেছে। সে কত আয় করেছে? [SOLUTION] =

Era

প্রথমে আমাদের বোঝা দরকার যে, অপর্ণা প্রতিঘন্টায় ১২ টাকা আয় করে।
অর্থাৎ, এক ঘন্টা সমান হলো ৬০ মিনিট।
সুতরাং, প্রতি মিনিটে তার আয় হচ্ছে ১২ টাকা / ৬০ মিনিট = ০.২ টাকা।
কিন্তু, গতকাল তিনি শুধু মাত্র ৫০ মিনিট বাচ্চা দেখাশোনা করেছেন।
তাই, তার গতকালের আয় হচ্ছে ০.২ টাকা/মিনিট * ৫০ মিনিট = ১০ টাকা।
সুতরাং, অপর্ণা গতকাল ১০ টাকা আয় করেছেন।

Enter your message...

+

Run

Playground messages can be viewed by anyone at your organization using the API.

Bidyarthi

Name

Bidyarthi

asst_uT1zLp58Kx3rR8a8IQ8HaPu

Instructions

You are an expert Bengali Math teacher who is very skilled at solving high school math word problems in Bengali. You will be given a Bengali [MATH] where you need to find just the [EQUATION] in Bengali. Your response format should be as

Model

gpt-4

TOOLS

Functions

Code interpreter

Retrieval

Updated 3/26, 10:33 AM

THREAD thread_G5ZfvaJHB1P2vA24zGvcE01v

369 tokens

Logs

User

Carefully Analyze the following Bangla word problem and provide the equation in Bengali of the [MATH] = একটি বিল্ডিংয়ে ১২টি কক্ষ আঁকার জন্য একজন চিত্রশিল্পীর প্রয়োজন হয়। প্রতিটি রুম রং করতে ৭ ঘন্টা সময় লাগে। যদি সে ইতিমধ্যেই ৫টি ঘর রং করে ফেলে, তাহলে বাকিটা রং করতে তার আর কত সময় লাগবে? [EQUATION] =

Bidyarthi

(১২ - ৫) * ৭

Enter your message...

+

Run

Playground messages can be viewed by anyone at your organization using the API.

Text (in Bengali)	Text (in English)
<p>Ques:এপ্রিলে নয়ন তার ৪৮ জন বন্ধুকে ক্লিপ বিক্রি করেছিলেন এবং তারপরের যে মাসে সে অর্ধেক ক্লিপ বিক্রি করেছিলেন। এপ্রিল এবং যে মাসে মোট কতটি ক্লিপ নয়ন বিক্রি করেছিলেন?</p>	<p>Ques: Nayan sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Nayan sell altogether in April and May?</p>
<p>Ans:(Correct) এপ্রিলে নয়ন তার ৪৮ জন বন্ধুকে ক্লিপ বিক্রি করেছেন। এবং তারপরের যে মাসে সে অর্ধেক ক্লিপ বিক্রি করেছেন। সাধারণ হিসাবে এপ্রিল এবং যে মাসে মোট কতটি ক্লিপ নয়ন বিক্রি করেছিলেন? এপ্রিলে নয়ন ৪৮ ক্লিপ বিক্রি করেছেন। এবং তারপরের মাসে = ২৪ টি ক্লিপ বিক্রি করেছেন। সাধারণ হিসাবে এপ্রিল এবং যে মাসে মোট কতটি ক্লিপ নয়ন বিক্রি করেছেন = $48 + 24 = 72$ টি। অতএব, এপ্রিল এবং যে মাসে মোট ৭২ টি ক্লিপ নয়ন বিক্রি করেছেন।</p>	<p>Ans:(Correct) In April, Nayan sold clips to 48 of his friends. And in the following month, he sold half as much clips. Therefore, the total number of clips Nayan sold in April and the following month is: April = 48 clips, Next month = 24 clips. Total = $48 + 24 = 72$ clips. So, Nayan sold a total of 72 crepes in April and the following month.</p>
<p>Ques:জুলি ১২০ পৃষ্ঠার একটি বই পড়ছে। গতকাল সে ১২ পৃষ্ঠা পড়তে পেরেছিল এবং আজ সে গতকালের দ্বিগুণ পৃষ্ঠা পড়েছে। সে যদি আগামীকাল বাকি পৃষ্ঠার অর্ধেক পড়তে চায়, তাহলে তাকে কত পৃষ্ঠা পড়তে হবে?</p>	<p>Ques: Julie is reading a book of 120 pages. Yesterday she read 12 pages and today she read twice as many pages as she did yesterday. If she wants to read half of the remaining pages tomorrow, how many pages does she need to read?</p>
<p>Ans:(Incorrect) জুলি ১২০ পৃষ্ঠার একটি বই পড়ছে। গতকাল সে ১২ পৃষ্ঠা পড়তে পেরেছিল এবং আজ সে গতকালের দ্বিগুণ পৃষ্ঠা পড়েছে। আমরা জানি, বইটির মোট পৃষ্ঠা সংখ্যা হলো ১২০ টি। গতকাল পড়া পৃষ্ঠা সংখ্যা হলো ১২ টি। আর আজ পড়া পৃষ্ঠা সংখ্যা হলো গতকালের দ্বিগুণ পৃষ্ঠা, অর্থাৎ $2 * 12 = 24$ টি। যদি আগামীকাল বাকি পৃষ্ঠার অর্ধেক পড়তে চায়, তাহলে তাকে কত পৃষ্ঠা পড়তে হবে? বইটির মোট পৃষ্ঠা সংখ্যা হলো ১২০ টি। গতকাল পৃষ্ঠা পড়া হয়েছে ১২ টি। আগামীকাল পড়া পৃষ্ঠার অর্ধেক পড়তে চান, তাহলে পড়তে হবে $(120 - 12) / 2 = 54$ টি পৃষ্ঠা। তাই, আগামীকাল বাকি পৃষ্ঠার অর্ধেক পড়তে হবে ৫৪ টি।</p>	<p>Ans:(Incorrect) Julie is reading a book of 120 pages. Yesterday she read 12 pages and today she read twice as many pages as she did yesterday. We know the total number of pages in the book is 120. Yesterday she read 12 pages. Today she read twice as many pages as yesterday, that is, $2 * 12 = 24$ pages. If she wants to read half of the remaining pages tomorrow, how many pages does she need to read? The total number of pages in the book is 120. Yesterday, she read 12 pages. If she wants to read half of the remaining pages tomorrow, she will have to read $(120 - 12) / 2 = 54$ pages. Therefore, she needs to read 54 pages tomorrow.</p>
<p>Ques:একটি ব্যবসায়িক লেনদেনের লাভ যথাক্রমে ২:৫ অনুপাতে ২ ব্যবসায়িক অংশীদার, মালেক এবং জিসানের মধ্যে ভাগ করা হয়। জিসান যদি ১২৫০০ পায়, তাহলে তাহলে মালেকের শার্ট কেনার পর (যার দাম ২০০ টাকা) কত টাকা থাকবে?</p>	<p>Ques: In a business transaction, the profit is divided in the ratio of 2:5 between two business partners, Malek and Jisan. If Jisan gets 2500, how much money will Malek have left after buying a shirt worth 200 takas?</p>
<p>Ans:(Incorrect) প্রবলেমটি ক্লিন্যাস করতে যদি মালেকের শার্ট না কেনা হয় তবে ভাগাভাগির অনুপাত বর্ণনা করে জিসান পাওয়া টাকার ফর্মুলা লিখা যেতে পারে। লাভের অনুপাত ২:৫। লাভের পরিমাণ x হলে মালেকের দাম (শার্টের দাম) = $2/2+5 * x$ এখন জিসান ১২৫০০ পায় তাই $x = 2500$ তাহলে মালেকের শার্টের দাম = $2/2+5 * 2500 = 2/7 * 2500 = 2*2500/7 = 5000/7$ তাই মালেকের শার্ট কেনার পর থাকা টাকা = $5000/7$</p>	<p>Ans:(Incorrect) To frame the problem, if Malek doesn't buy the shirt, we can write the formula for the amount Jisan gets based on the sharing ratio. The profit ratio is 2:5. If the profit amount is x, then Malek's share (shirt price) = $2/(2+5) * x$. Now Jisan gets 2500, so $x = 2500$. Then Malek's share is $= 2/(2+5) * 2500 = 2/7 * 2500 = 2*2500/7 = 5000/7$ So, the amount left with Malek after buying the shirt = $5000/7$.</p>

Figure A.1: Example of correct chains of thought produced by the GPT-3.5 for the BGSM8K dataset for Few Shot

Text (in Bengali)	Text (in English)
<p>Ques: এপ্রিলে নয়ন তার ৪৮ জন বন্ধুকে ক্লিপ বিক্রি করেছিল এবং তারপরে মে মাসে সে অর্ধেক ক্লিপ বিক্রি করেছিল। এপ্রিল এবং মে মাসে মোট কতটি ক্লিপ নয়ন বিক্রি করেছিল?</p>	<p>Ques: Nayan sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Nayan sell altogether in April and May?</p>
<p>Ans:(Correct) মে মাসে তিনি শুধু $48/2 = <<48/2=24>>24$ টি ক্লিপ বিক্রি করেছিলেন। এপ্রিল এবং মে মাসে তিনি মোট $48 + 24 = <<48+24=72>>72$ টি ক্লিপ বিক্রি করেছিলেন।</p>	<p>Ans:(Correct) In May, he sold only $48/2 = <<48/2=24>>24$ clips. Therefore, in April and May, he sold a total of $48 + 24 = <<48+24=72>>72$ clips.</p>
<p>Ques: মাখনের একটি ফুলের বাগান আছে যেখানে তিন ধরনের রঙের ফুল রয়েছে। সেখানে দশটি হলুদ ফুল আছে এবং বেগুনি রঙের ফুল তার ৮০% বেশি আছে। হলুদ এবং বেগুনি ফুলের মোট সংখ্যার মাত্র ২৫% সবুজ ফুল আছে। মাখনের বাগানে কতটি ফুল আছে?</p>	<p>Ques: Makhan has a flower garden with three types of colored flowers. There are ten yellow flowers and 80% more purple flowers than yellow. There are only 25% green flowers of the total number of yellow and purple flowers. How many flowers are in Makhan's garden?</p>
<p>Ans:(Incorrect) বেগুনি ফুলের সংখ্যা $10 + (10 \times 80/100) = <<10+(10*80/100)=18>>18$ টি। হলুদ এবং বেগুনি ফুলের সংখ্যা $10 + 18 = <<10+18=28>>28$ টি। সবুজ ফুল $28 \times 8 = <<28*8=224>>224$ টি। সুতরাং, মাখনের বাগানে মোট $112 + 28 = <<112+28=140>>140$ টি ফুল রয়েছে।</p>	<p>Ans:(Incorrect) The number of purple flowers is $10 + (10 \times 80/100) = <<10+(10*80/100)=18>>18$. The number of yellow and purple flowers is $10 + 18 = <<10+18=28>>28$. Green flowers are $28 \times 4 = <<28*4=112>>112$. Therefore, Makhan's garden has a total of $112 + 28 = <<112+28=140>>140$ flowers.</p>
<p>Ques: অপর্ণা বাচ্চা দেখাশোনা করার জন্য ঘন্টায় ১২ টাকা আয় করে। গতকাল সে মাত্র ৫০ মিনিট বাচ্চা দেখাশোনা করেছে। সে কত আয় করেছে?</p>	<p>Ques: Aparna earns 12 takas per hour for babysitting. Yesterday she babysat for only 50 minutes. How much did she earn?</p>
<p>Ans:(Correct) গতকাল অপর্ণা মাত্র $50/60 = <<50/60=0.83>>0.83$ (ঘন্টা মাপে পরিবর্তিত) ঘন্টা বাচ্চা দেখাশোনা করেছে। সে ঘন্টায় ১২ টাকা আয় করে, সুতরাং সে মোট $12 \times 0.83 = 9.96$ টাকা আয় করেছে। তবে মূল্যটি পূর্ণ সংখ্যা হওয়া উচিত, তাই উপরিস্থিতি মূল্য ১০ টাকা হবে।</p>	<p>Ans:(Correct) Yesterday, Aparna babysat for only $50/60 = <<50/60=0.83>>0.83$ hours (converted to hours). She earns 12 takas per hour, so she earned a total of $12 \times 0.83 = 9.96$ taka. However, the amount should be rounded, so the final amount is 10 taka.</p>

Figure A.2: Example of correct chains of thought produced by the GPT-4o for the BGSM8K dataset for Few Shot

Text (in Bengali)	Text (in English)
<p>Ques: এপ্রিলে নয়ন তার ৪৮ জন বন্ধুকে ক্লিপ বিক্রি করেছিল এবং তারপরে মে মাসে সে অর্ধেক ক্লিপ বিক্রি করেছিল। এপ্রিল এবং মে মাসে মোট কতটি ক্লিপ নয়ন বিক্রি করেছিল?</p>	<p>Ques: Nayan sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Nayan sell altogether in April and May?</p>
<p>Ans:(Correct) মে মাসে তিনি শুধু $48/2 = \ll 48/2=24 \gg$ ২৪টি ক্লিপ বিক্রি করেছিলেন। এপ্রিল এবং মে মাসে তিনি মোট $48 + 24 = \ll 48+24=72 \gg$ ৭২টি ক্লিপ বিক্রি করেছিলেন।</p>	<p>Ans:(Correct) In May, he sold only $48/2 = \ll 48/2=24 \gg$ 24 clips. Therefore, in April and May, he sold a total of $48 + 24 = \ll 48+24=72 \gg$ 72 clips.</p>
<p>Ques: মাখনের একটি ফুলের বাগান আছে যেখানে তিন ধরনের রঙের ফুল রয়েছে। সেখানে দশটি হলুদ ফুল আছে এবং বেগুনি রঙের ফুল তার ৮০% বেশি আছে। হলুদ এবং বেগুনি ফুলের মোট সংখ্যার মাত্র ২৫% সবুজ ফুল আছে। মাখনের বাগানে কতটি ফুল আছে?</p>	<p>Ques: Makhan has a flower garden with three types of colored flowers. There are ten yellow flowers and 80% more purple flowers than yellow. There are only 25% green flowers of the total number of yellow and purple flowers. How many flowers are in Makhan's garden?</p>
<p>Ans:(Incorrect) বেগুনি ফুলের সংখ্যা $10 + (10 \times 80/100) = \ll 10+(10*80/100)=18 \gg$ ১৮ টি। হলুদ এবং বেগুনি ফুলের সংখ্যা $10 + 18 = \ll 10+18=28 \gg$ ২৮ টি। সবুজ ফুল $28 \times 8 = \ll 28*8=224 \gg$ ২২৪ টি। সুতরাং, মাখনের বাগানে মোট $112 + 28 = \ll 112+28=140 \gg$ ১৪০ টি ফুল রয়েছে।</p>	<p>Ans:(Incorrect) The number of purple flowers is $10 + (10 \times 80/100) = \ll 10+(10*80/100)=18 \gg$ 18. The number of yellow and purple flowers is $10 + 18 = \ll 10+18=28 \gg$ 28. Green flowers are $28 \times 4 = \ll 28*4=112 \gg$ 112. Therefore, Makhan's garden has a total of $112 + 28 = \ll 112+28=140 \gg$ 140 flowers.</p>
<p>Ques: অপর্ণা বাচ্চা দেখাশোনা করার জন্য ঘন্টায় ১২ টাকা আয় করে। গতকাল সে মাত্র ৫০ মিনিট বাচ্চা দেখাশোনা করেছে। সে কত আয় করেছে?</p>	<p>Ques: Aparna earns 12 takas per hour for babysitting. Yesterday she babysat for only 50 minutes. How much did she earn?</p>
<p>Ans:(Correct) গতকাল অপর্ণা মাত্র $50/60 = \ll 50/60=0.83 \gg$ ০.৮৩ (ঘন্টা মাপে পরিবর্তিত) ঘন্টা বাচ্চা দেখাশোনা করেছে। সে ঘন্টায় ১২ টাকা আয় করে, সুতরাং সে মোট ১২ টাকা * ০.৮৩ = ৯.৯৬ টাকা আয় করেছে। তবে মূল্যটি পূর্ণ সংখ্যা হওয়া উচিত, তাই উপরলিখিত মূল্য ১০ টাকা হবে।</p>	<p>Ans:(Correct) Yesterday, Aparna babysat for only $50/60 = \ll 50/60=0.83 \gg$ 0.83 hours (converted to hours). She earns 12 takas per hour, so she earned a total of $12 \text{ taka} * 0.83 = 9.96 \text{ taka}$. However, the amount should be rounded, so the final amount is 10 taka.</p>

Figure A.3: Example of correct chains of thought produced by the GPT-3.5 for the BGSM8K dataset for Fine Tuning

Appendix B

Publication

A conference work from our research titled, "**Empowering Bengali Education with AI: Solving Bangla Math Word Problems through Transformer Models**" has been accepted and presented in the 27th INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION TECHNOLOGY (ICCIT 2024)

Generated using Undergraduate Thesis L^AT_EX Template, Version 1.4. Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh.

This thesis was generated on Monday 2nd June, 2025 at 3:47pm.