Problem Statement

- Introducing Generative AI (GenAI) concepts and performing simple inference with a Large Language Model (LLM) on a CPU.

- Fine-tuning an LLM using a math dataset to develop a custom chatbot capable of solving mathematical problems in conversational contexts. This entails optimizing the fine-tuning process to make it accessible and efficient for researchers and developers.

Unique Idea Brief (Solution)

Dataset Preparation:

Problem Data Preparation:

- Each problem instruction and solution is encapsulated between `<s>` and `</s>` tags.
- The problem statement is followed by a solution explanation.

Processing Each Problem:

- Iterate over each `<s>` ... `</s>` block.
- Extract the problem instruction and the solution explanation.

Preprocessing Steps:

- Strip any leading or trailing whitespace from the extracted text.
- Split the text into "instruction" and "response" using a specific delimiter or structure (`<s>` ... `</s>`).
- Print or further process each pair as required.

Model Preparation:

1)Package Installation: Utilizes pip to install specific versions of Python packages:

- `accelerate==0.21.0`: Optimizes deep learning model training and evaluation.
- `peft==0.4.0`: Facilitates efficient fine-tuning of large language models.

- `bitsandbytes==0.40.2`: Manages GPU memory and optimizes operations.
- `transformers==4.31.0`: Popular for NLP tasks, including model training and inference.
- `trl==0.4.7`: Integrates reinforcement learning with transformer-based models.

2)Technologies Used for LLM:

- **OS**: Interacts with the operating system.
- **torch**: PyTorch library for tensor computations and model building.
- **load_dataset**: Loads and preprocesses datasets from Hugging Face's datasets library.
- **AutoModelForCausalLM, AutoTokenizer**: Automatically loads pretrained language models and tokenizers.
- **BitsAndBytesConfig**: Manages GPU memory and operations efficiently.
- **HfArgumentParser, TrainingArguments**: Handles training script arguments and configurations.

- **pipeline**: High-level API for model inference.
- **logging**: Provides debugging and logging capabilities.
- **LoraConfig, PeftModel, SFTTrainer**: Configurations and classes for fine-tuning models with LoRA and reinforcement learning techniques.

3)Fine-Tuning Configuration:

- Fine-tunes "NousResearch/Llama-2-7b-chat-hf" using LoRA for efficiency.
- Implements 4-bit quantization with bitsandbytes to reduce memory usage.

4)Training Process:

- Configures training parameters like batch size, learning rate, and optimizer settings.
- 2198 steps over one epoch with an average loss of approximately 0.998.
- Training duration of about 6072 seconds (1 hour 41 minutes).
- Processes 1.448 samples per second, indicating training efficiency.
- Runs training with metrics monitoring and efficiency enhancements.

5)Dataset Handling:

- Loads datasets from Hugging Face hub for training.

6)Memory Management:

- Configures GPU memory usage with 4-bit quantization and fp16 precision.

7)Model Initialization:

- Loads pretrained models with specified configurations.
- Sets up tokenizers and manages padding for input sequences.

8)LoRA Configuration:

- Initializes LoRA with alpha scaling, dropout, and rank parameters.

9)Training Parameter Setup:

- Sets up directories for saving models and checkpoints.
- Configures learning rate schedules and gradient handling.

10)Training Execution:

- Initiates training process using SFTTrainer with configured parameters.

11)Training Metrics:

- Tracks training efficiency with metrics like training time, samples per second, and loss.

12)Model Saving and Logging:

- Saves fine-tuned models and manages logging verbosity.

13)Inference Setup:

- Sets up environment for generating text with fine-tuned models.

14)Model Reload and Memory Management:

- Clears memory, reloads models in different precisions, and manages garbage collection.

15)Hugging Face CLI Operations:

- Logs into Hugging Face CLI and pushes models/tokenizers to the Model Hub.

OUTPUT:

- The code initializes a text generation pipeline using a pretrained model and tokenizer. It processes a user-provided prompt, generates a response up to 200 tokens long, and prints the resulting text.

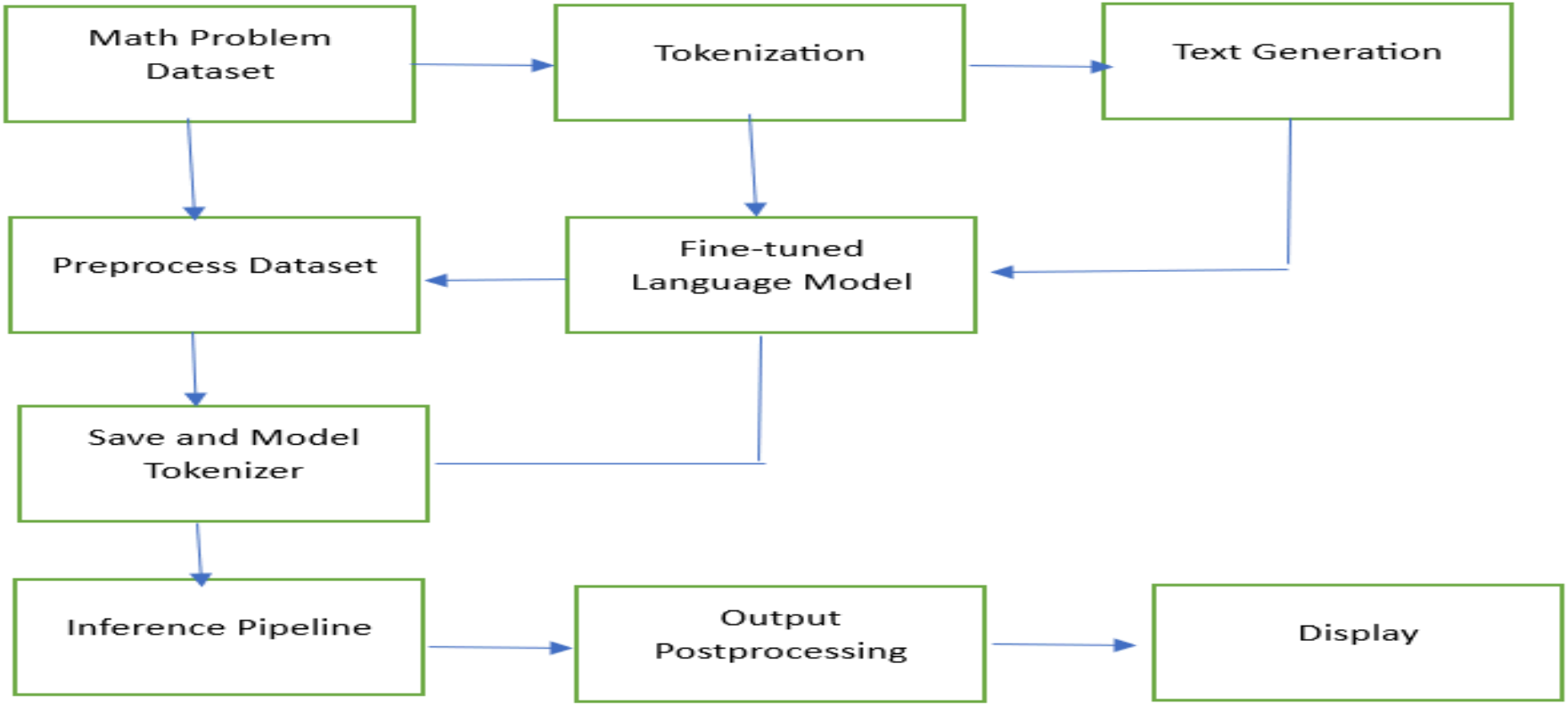
- This demonstrates using AI to generate responses based on specific prompts, enhancing interactive capabilities in applications such as automated Q&A systems, virtual assistants, and educational tools.

Features Offered

- **Efficient Fine-Tuning**: Streamlines parameter-efficient fine-tuning of models, reducing computational overhead.
- **Optimized Training**: Utilizes mixed precision and distributed training for faster convergence and reduced memory usage.
- **Reinforcement Learning Integration**: Integrates reinforcement learning techniques, enabling sophisticated training strategies.
- **Memory Management**: Efficient GPU memory handling and optimization for training large-scale models.
- **High-Level APIs**: Simplifies deployment and inference pipelines, facilitating easy integration into applications.
- **Custom Model Handling**:Supports custom model configurations and optimizations like Low-Rank Adaptation (LoRA).
- **Logging and Debugging**:Provides tools for monitoring training progress, debugging, and analyzing model performance metrics.

Process flow

- **Math Problem Dataset**: Initial dataset containing math problem instructions and solutions.
- **Tokenization**: Process of converting text into tokens suitable for model input.
- **Fine-tuned Language Model**: Model trained on the preprocessed dataset to generate solutions.
- **Inference Pipeline**: Takes user prompts, tokenizes them, feeds them into the model, and generates text responses.
- **Output Post processing**: Ensures generated solutions are formatted correctly and checked for coherence.
- **Display**: Presents the final generated solutions to the user in a readable format


Math Problem Dataset
Tokenization
Text Generation
Preprocess Dataset
Fine-tuned Language Model
Save and Model Tokenizer
Inference Pipeline
Output Postprocessing
Display

Architecture Diagram

Math Problem Dataset:

- Initial dataset containing math problems and their solutions.

Preprocessing Module:

- Formats and cleans the dataset.
- Tokenizes data for input into the fine-tuned language model.

Fine-tuned Language Model:

- Trained on the preprocessed math problem dataset.
- Capable of understanding and generating text based on input prompts.

Model Input Module:

- Prepares token IDs, mask IDs, and segment IDs required by the language model for inference.

Model Inference & Generation:

- Processes tokenized input to generate text.
- Generates solutions to math problems based on input queries.

Generated Solutions Module:

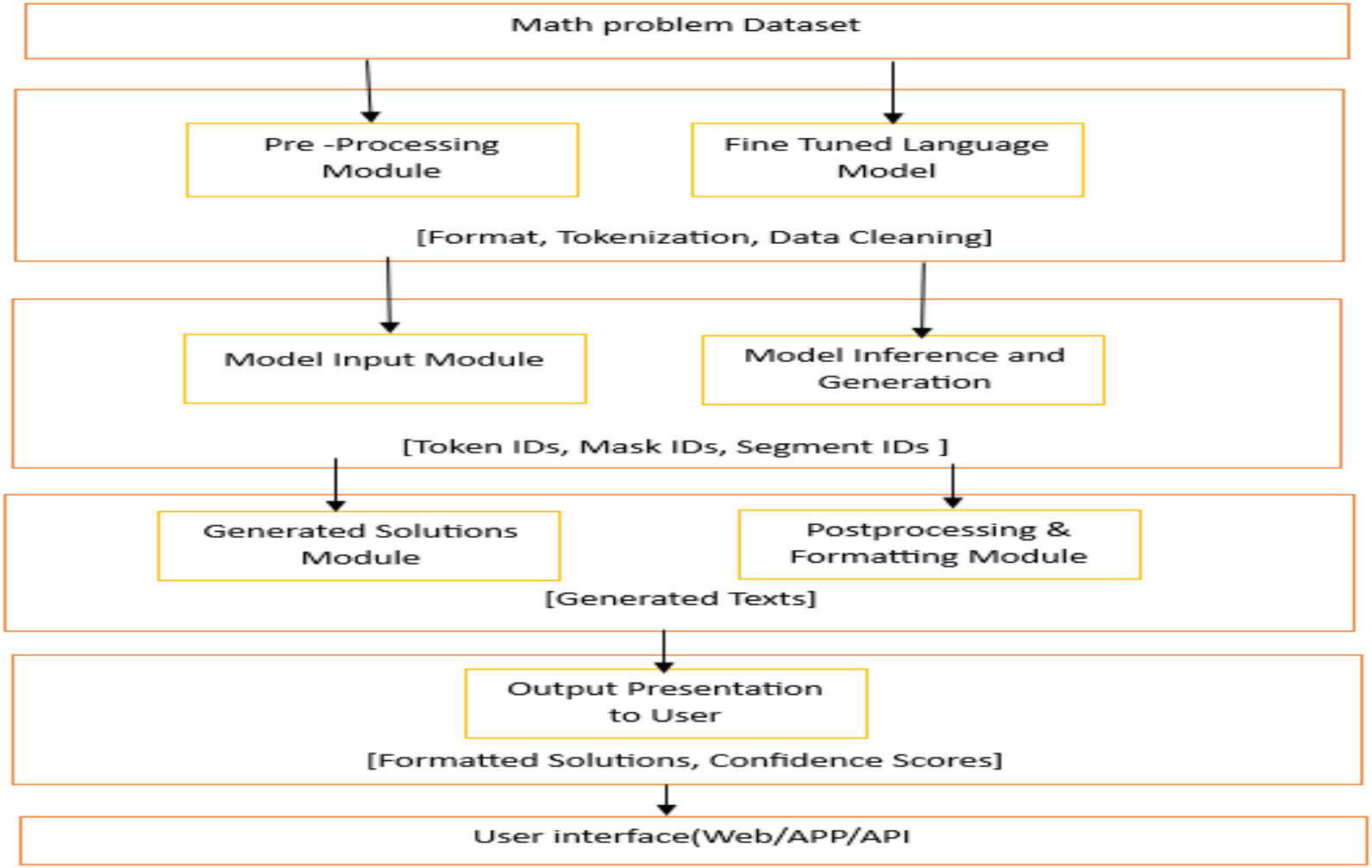
- Receives and processes generated text from the language model.
- Performs postprocessing tasks such as formatting and ensuring coherence.

Output Presentation:

- Presents formatted solutions and confidence scores to the user.
- Interfaces with the user through web interfaces, applications, or APIs.

User Interface:

Web interface, mobile app, or API endpoint where users input queries and receive generated solutions.


Math problem Dataset
Pre -Processing Module
Fine Tuned Language Model
[Format, Tokenization, Data Cleaning]
Model Input Module
Model Inference and Generation
[Token IDs, Mask IDs, Segment IDs]
Generated Solutions Module
Postprocessing & Formatting Module
[Generated Texts]
Output Presentation to User
[Formatted Solutions, Confidence Scores]
User interface(Web/APP/API

Technologies used

PyTorch (torch):

- Used for tensor computations and building deep learning models.
- Supports efficient GPU acceleration and automatic differentiation.

Hugging Face Transformers (`transformers`):

- Popular library for natural language processing (NLP) tasks.
- Provides pretrained models, tokenizers, and tools for model training and inference.

Accelerate (`accelerate`):

- From Hugging Face, optimizes deep learning model training and evaluation.
- Facilitates distributed training, mixed precision, and other performance optimizations.

PEFT (`peft` - Parameter-Efficient Fine-Tuning):

- Designed for efficient fine-tuning of large language models.
- Reduces the number of parameters updated during training to improve efficiency.

BitsAndBytes (`bitsandbytes`):

- Manages GPU memory efficiently.
- Optimizes operations for training and inference of deep learning models, particularly those with large-scale parameters.

TRL (`trl` - Transformers Reinforcement Learning):

- Integrates reinforcement learning techniques with transformer-based models.
- Enables advanced training strategies and fine-tuning approaches.

Hugging Face Datasets (`load_dataset`):

- Simplifies loading and preprocessing datasets from various sources.
- Provides datasets for training and evaluation in NLP tasks.

AutoModelForCausalLM and AutoTokenizer:

- Automatically loads pretrained causal language models and their corresponding tokenizers.
- Simplifies model setup for tasks like text generation.

BitsAndBytesConfig:

- Configuration class for efficient GPU memory management and operations.
- Supports optimizations like 4-bit quantization to reduce memory usage while maintaining performance.

HfArgumentParser and TrainingArguments:

- Hugging Face tools for handling command-line arguments specific to training scripts.
- Configures training parameters such as batch size, learning rate, and optimizer settings.

SFTTrainer:

- Trainer class from TRL library for supervised fine-tuning of transformer models.
- Manages the training process with specified datasets, models, and training arguments.

Logging:

- Provides capabilities for tracking and debugging the model training and evaluation process and tracking training metrics.

Team members and contribution:

Thissyakkanna S M: Finetuned the Llama-2-7b-chat-hf model for Math Dataset and created a custom chatbot.

Jayadhanush R: Finetuned the Llama-2-7b-chat-hf model for Math Dataset and created a custom chatbot.

Prenitha S P: preparation and preprocessing of Math dataset for the model, Documentation Works.

Conclusion:

- The model processes math problem instructions and generates detailed solutions. It handles a variety of mathematical concepts, including arithmetic, algebra, geometry, probability, and physics.
- Each problem instruction is paired with a clear, step-by-step solution that includes calculations, formulas, and explanations where applicable. The model ensures clarity and accuracy in its responses, aiding users in understanding and solving complex mathematical problems across different domains.
- This model could be particularly useful in educational settings, tutoring platforms, or any scenario where automated assistance with mathematical problem-solving is required.
- It leverages natural language processing to interpret problem statements and generate coherent, structured solutions that guide users through each step of the problem-solving process.