Enhancing Mental Health Support through Conversational AI: Fine-tuning google-t5-base model mental health conversational dataset

Rohit Sharma, Jitender Singh, Rahul Kumar
M.Tech Student CICPS, Indian Institute of Technology Guwahati
rohit.sharma@iitg.ac.in, s.jitender@iitg.ac.in, rahul.k5027@iitg.ac.in

***Abstract*—Language models have become increasingly popular for various natural language processing tasks, including chatbot applications. In this project, we fine-tuned a pre-trained language model using a conversational dataset focused on mental health counseling conversations. The dataset was preprocessed, tokenized, and used to train the language model. We employed techniques such as data loading, tokenization using Hugging Face Transformers, and model fine-tuning with PyTorch Lightning. The trained model was evaluated using metrics such as BLEU score and perplexity to assess its performance. Our results show promising outcomes in generating responses for mental health-related prompts, demonstrating the potential of fine-tuned language models for chatbot applications in sensitive domains. This project contributes to the advancement of conversational AI systems tailored for mental health support and counseling.**

***Index Terms*—Language models, fine-tuning, chatbot, mental health, conversational dataset**

I. Introduction

NATURAL language processing (NLP) has witnessed significant advancements in recent years, fueled by the development of large-scale pre-trained language models. These models, such as GPT (Generative Pre-trained Transformer) and T5 (Text-To-Text Transfer Transformer), have demonstrated remarkable capabilities in understanding and generating human-like text across a wide range of tasks. Among the various applications of NLP, chatbots have gained prominence for their ability to engage in natural language conversations and provide assistance or information on diverse topics.

Chatbots have been employed in various domains, including customer service, education, and healthcare. In the field of mental health, chatbots have shown potential as a scalable and accessible means of providing support and counseling services to individuals experiencing psychological distress. However, building effective chatbots for mental health applications poses several challenges, including the need for empathy, sensitivity to context, and the ability to generate appropriate responses tailored to the user's emotional state.

To address these challenges, researchers have explored the use of fine-tuned language models for chatbot development. Fine-tuning involves training a pre-trained language model on domain-specific data to adapt it to a particular task or domain. By fine-tuning on a dataset of mental health counseling conversations, we aim to develop a chatbot capable of providing empathetic and contextually relevant responses to users seeking mental health support.

In this project, we focus on fine-tuning a pre-trained language model using a conversational dataset comprising interactions between mental health counselors and clients. Our objective is to leverage the rich conversational data to train a language model that can understand and generate responses reflective of the nuanced nature of mental health conversations. We employ state-of-the-art techniques in data preprocessing, tokenization, and model training to optimize the performance of the fine-tuned language model.

The remainder of this report is organized as follows: Section II provides a detailed overview of the preprocessing steps undertaken to prepare the dataset for fine-tuning. Section III discusses the methodology employed for fine-tuning the language model and the evaluation metrics used to assess its performance. Section IV presents the results of our experiments and discusses their implications. Finally, Section V concludes the report and outlines potential avenues for future research in the field of chatbot development for mental health support.

A. Objective

The primary objective of this project is to develop a chatbot tailored for mental health support and counseling by fine-tuning a pre-trained language model on a dataset of mental health counseling conversations. Specifically, our goals are as follows:

1. Preprocess the conversational dataset to extract relevant information and prepare it for fine-tuning.
2. Fine-tune a pre-trained language model using the preprocessed dataset to adapt it to the domain of mental health counseling.
3. Evaluate the performance of the fine-tuned model using BLEU score metrics.
4. Analyze the generated responses to assess the chatbot's ability to provide empathetic and contextually relevant support to users.
5. Explore potential enhancements and future directions for improving the chatbot's performance and usability in real-world applications.

II. Preprocessing

In this section, we discuss the preprocessing steps carried out on the dataset for further analysis and model training.

A. Tokenization Analysis

Tokenization is a crucial step in natural language processing (NLP) tasks, where text is split into individual tokens or words for analysis. We employed the 'FindingTokensAnalyze' class to analyze the tokenization statistics of the dataset. This class loads a specified tokenizer and calculates various statistics, including the minimum, maximum, median, and 95th percentile token lengths.

B. Text-to-Text Data Preparation

The 'TextToTextData' class is responsible for loading the raw dataset and converting it into a structured format suitable for training models. It loads samples from a JSON file, where each sample consists of a source text and a corresponding target text. These samples are then processed and saved as instances of the 'TextToTextSample' class.

C. Mental Health Data Analysis

The 'Mental Data' class is designed specifically for preprocessing a dataset related to mental health counseling conversations. It loads samples from a JSON file containing context-response pairs and analyzes various aspects of the data, such as the total number of samples and the proportion of samples with context information.

D. Text-to-Text Dataset Creation

Finally, the 'TextToTextDataset' class creates a PyTorch dataset for training a text-to-text model. It tokenizes the source and target texts using a specified tokenizer and formats the data into dictionary format suitable for model input. Additionally, it provides a method for converting raw dictionary data into human-readable format for analysis purposes.

Overall, these preprocessing steps ensure that the dataset is properly formatted and tokenized for training and analysis tasks.

III. Model Architecture: Google T5 Base

The Google T5 base model is a variant of the Text-To-Text Transfer Transformer (T5), developed by Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. The T5 model introduces a unified text-to-text framework for various Natural Language Processing (NLP) tasks.

A. Model Description

The T5 model reframes all NLP tasks into a text-to-text format, where both the input and output are represented as text strings. This approach contrasts with BERT-style models, which typically output either a class label or a span of the input. By adopting a text-to-text format, T5 enables the use of the same model, loss function, and hyperparameters across different NLP tasks.

B. Key Features

- **Unified Framework**: T5 adopts a unified framework where all NLP tasks are represented as text-to-text transformations.
- **Input-Output Consistency**: Both input and output are treated as text strings, providing consistency across tasks.
- **Parameter Size**: The T5-Base checkpoint consists of 220 million parameters.
- **Multilingual Support**: The T5 model supports multiple languages, including English, French, Romanian, and German.

C. Usage and Resources

- **Research Paper**: The development and architecture of T5 are detailed in the associated research paper.
- **Google's T5 Blog Post**: Google has published a blog post providing insights into the T5 model and its applications.
- **GitHub Repository**: The T5 model's source code and implementation details are available on the associated GitHub repository.

D. License

The T5 model is licensed under the Apache 2.0 license, allowing for open usage and modification.

E. Related Models

The T5 model is part of a series of checkpoints, including various sizes and configurations. Other T5 checkpoints are available for different tasks and applications.

IV. Fine-tuning Model

In this section, we discuss the fine-tuning process carried out to adapt a pre-trained model to a specific task or dataset.

A. Argument Parsing

The 'get args' function parses command-line arguments to configure the fine-tuning process. It allows specifying parameters such as the model name or path, maximum source and target sequence lengths, data path, training epochs, batch size, learning rate, seed, and output directory.

B. Lightning Model Definition

The 'LightningModel' class defines the PyTorch Lightning module for fine-tuning. It inherits from 'pl.LightningModule' and initializes the model architecture and tokenizer. The model architecture is loaded from a pre-trained checkpoint specified by the 'model name or path' parameter. Optionally, the model can be augmented with techniques such as LoRA (Low-Rank Adaptation) or gradient checkpointing.

C. Model Forward Pass

The 'forward' method defines the forward pass of the model. It takes input sequences, attention masks, decoder input sequences, and labels as input and returns the model outputs, including logits and loss.

D. Training Step

The ‘training step‘ method defines a single training step for the model. It calculates the loss using the ‘ step‘ method and logs the loss value for monitoring during training.

E. Optimizer Configuration

The ‘configure optimizers‘ method configures the optimizer for training. It defines the optimizer algorithm (Adafactor) and specifies the learning rate and weight decay parameters.

F. Data Loading

The ‘train dataloader‘ method creates the data loader for training. It loads the dataset using the ‘TextToTextDataset‘ class, which tokenizes the input and target sequences and formats them into batches for efficient training.

G. Main Function

The ‘main‘ function orchestrates the fine-tuning process. It initializes the model, sets up callbacks for model checkpointing, defines the training strategy (e.g., using FSDP for model parallelism), and trains the model using the PyTorch Lightning Trainer.

Overall, these steps enable the fine-tuning of a pre-trained model on a specific task or dataset, leading to improved performance and adaptation to the target domain.

V. Evaluation of Language Model (LLM)

In this section, we discuss the evaluation process for the Language Model (LLM) fine-tuned on a specific task or dataset.

A. Evaluation Dataset

The evaluation dataset is loaded using the ‘TextToTextDataset‘ class, which prepares the input-output pairs for evaluation. It tokenizes the input sequences and formats them into batches for efficient evaluation.

B. Response Generation

A portion of the evaluation dataset is used only because of unavailability of computing resource. Responses are generated using the fine-tuned LLM. For each input sequence in the evaluation dataset, the model generates a response sequence using beam search decoding.

C. BLEU Score Computation

The BLEU (Bilingual Evaluation Understudy) score is a metric commonly used to evaluate the quality of generated text. It measures the similarity between the generated responses and the reference target texts. The BLEU score is computed using the ‘corpus bleu‘ function from the NLTK library.

D. BLEU Score Formula

The BLEU score is calculated as the geometric mean of modified precision scores for n-grams of different lengths. The modified precision score for each n-gram is the ratio of the number of n-grams in the generated text that appear in the reference text to the total number of n-grams in the generated text.

$$BLEU = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{1}$$

Where:

- BP is the Brevity Penalty, which penalizes short translations.
- w_n is the weight assigned to the precision of n-grams.
- p_n is the modified precision score for n-grams.

VI. Results

A. Effect of Learning Rate on BLEU Score

We conducted experiments to investigate the impact of varying learning rates on the BLEU score during the fine-tuning process. The experiments were performed with and without using LoRA parameters.

Learning Rate	BLEU Score
0.0025	0.2931
0.003	0.3336
0.004	0.2778
0.005	0.3140

TABLE I
BLEU Scores for Different Learning Rates (Without LoRA)

1) Without LoRA: The BLEU scores obtained without using LoRA ranged from 0.2778 to 0.3336, with the highest score achieved at a learning rate of 0.003.

Learning Rate	BLEU Score
0.0025	0.0818
0.003	0.0917
0.004	0.0989
0.005	0.1148

TABLE II
BLEU Scores for Different Learning Rates (With LoRA)

2) With LoRA (Rank=8, LoRA Alpha=8, LoRA Dropout=0.1): When using LoRA with parameters rank=8, LoRA Alpha=8, and LoRA Dropout=0.1, the BLEU scores ranged from 0.0818 to 0.1148. The highest score was achieved at a learning rate of 0.005.

B. Discussion

The results indicate that the choice of learning rate significantly affects the performance of the fine-tuned model. Without LoRA, a learning rate of 0.003 resulted in the highest BLEU score. However, when using LoRA, a higher learning rate of 0.005 led to the best performance.

Achieving higher BLEU scores suggests that the generated responses are more similar to the training data. However, excessively high BLEU scores may indicate overfitting, where the model memorizes the training data instead of learning to generalize to unseen data. Therefore, it is essential to strike a balance and avoid overly high BLEU scores to ensure better generalization performance.

VII. Conclusion

In this project, we explored the effectiveness of fine-tuning the Google T5 base model for a text-to-text transfer task. We began by preprocessing the data, including tokenization and data loading, to prepare it for fine-tuning. Subsequently, we employed the Lightning framework to finetune the T5 model on our dataset, utilizing various hyperparameters such as learning rate, batch size, and optimization algorithms.

Through our experiments, we observed the impact of different hyperparameters on the model's performance, particularly focusing on the BLEU score as a metric for evaluating the quality of generated responses. We investigated the influence of learning rate on BLEU scores, both with and without LoRA parameters. Our results demonstrated that the choice of learning rate significantly affects the model's performance, with higher rates leading to improved BLEU scores in some cases.

Furthermore, we discussed the implications of achieving high BLEU scores, emphasizing the balance between model performance and generalization capability. While higher BLEU scores suggest better alignment with the training data, excessively high scores may indicate overfitting, necessitating caution in model selection and hyperparameter tuning.

Overall, this project contributes to the understanding of fine-tuning large language models for text generation tasks and provides insights into optimizing hyperparameters for improved performance. Future work may involve exploring additional techniques for enhancing model robustness and generalization, such as ensemble methods or data augmentation strategies.

Acknowledgment

We would like to express our sincere gratitude to our course instructor, Dr. Chiranjib Sur, for their invaluable guidance and support throughout the duration of this project. Their expertise and encouragement have been instrumental in shaping our understanding of Natural Language Processing with Large Language Models (NLP with LLM).

Special thanks to our peers and colleagues for their constructive discussions and collaboration, which have enriched our learning experience.

Rohit Sharma, Jitender Singh, and Rahul Kumar M.Tech Students Center for Intelligent Cyber Physical Systems Indian Institute of Technology Guwahati

References

[1] Hugging Face Model Hub:
https://huggingface.co/google-t5/t5-base

[2] Hugging Face Datasets Hub:
https://huggingface.co/datasets/Amod/mental$_health_counseling_conversations$