National College of Ireland

# Configuration Manual: Utilizing Counselor-Client Dialogues to Develop a Memory-Efficient Mental Health Question-Answering System with Large Language Models

MSc Research Project
Artificial Intelligence

Diya Srivastava
Student ID: x23177608

School of Computing
National College of Ireland

Supervisor: Arundev Vamadevan

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Diya Srivastava |
| **Student ID:** | x23177608 |
| **Programme:** | Artificial Intelligence |
| **Year:** | 2025 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Arundev Vamadevan |
| **Submission Due Date:** | 12/12/2024 |
| **Project Title:** | Configuration Manual: Utilizing Counselor-Client Dialogues to Develop a Memory-Efficient Mental Health Question-Answering System with Large Language Models |
| **Word Count:** | 849 |
| **Page Count:** | 14 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | Diya Srivastava |
| **Date:** | 24th January 2025 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual: Utilizing Counselor-Client Dialogues to Develop a Memory-Efficient Mental Health Question-Answering System with Large Language Models

Diya Srivastava
x23177608

## 1 Introduction

This research develops a Mental Health Question-Answering system by leveraging Large Language Models(LLMs). The study experiments with 5 LLMs i.e. Flan-T5 Chung et al. (2022), Tiny-LlamaZhang et al. (2024), Llama-2 Touvron et al. (2023), Gemma-2 Team (2024) and GPT-Neo Black et al. (2021) utilizing Parameter Efficient fine-tuning technique LoRa and Memory Efficient strategy Quantization, thus Q-LoRa. All these models have been trained with the same configuration and parameters for a comparative research, Hence this configuration manual showcase the code artifacts for 1 model i.e. Gemma-2 that shall be followed for other 4 LLMs. The study initially trains the LLMs on 1 epoch scrutinizing the performance in least resources, and perform Re-training for 10 epochs on the best 3 promising models i.e. Llama-2, Gemma-2 and GPT-Neo. Further, the models are evaluated on their performance through evaluation Metrics such as ROUGE, BERT and BLEU score accompanied with Inference on different questions to scrutinize the generated responses qualitatively.

## 2 Configuration

### 2.1 Hardware Requirements

This project was conducted on 2020 MacBook M1 with 8GB of memory, running MacOS Sonoma version 14.3.1, and equiped with 245.11GB of internal storage. Further, the model training and evaluation was conducted in Google Colaboratory Pro, using the following Hardware accelerator (Compute Engine Backend GPU):

**T4:**

- System RAM: 12.7GB
- GPU RAM: 15.0 GB
- Disk: 235.7GB

**L4:**

- System RAM: 53.0GB
- GPU RAM: 22.5GB
- Disk: 235.7GB

**A100:**

- System RAM: 83.5GB
- GPU RAM: 40.0GB
- Disk: 235.7GB

### 2.2 Software and Libraries Requirements

The following Python Version and libraries have been used to perform this research:

- Python version 3.10.12
- bitsandbytes 0.400.2
- trl 0.4.7
- peft 0.4.0
- accelerate 0.21.0
- transformers 4.31.0
- datasets
- huggingface
- torch
- evaluate
- bert_score
- rouge_score
- json

## 3 Dataset Source

Dataset leveraged in this research is an open-source publicly available dataset on HuggingFace Jia Xu (2024):

**MentalChat16k:** `https://huggingface.co/datasets/ShenLab/MentalChat16K`

```
import os
import torch
from datasets import load_dataset, DatasetDict, Dataset, concatenate_datasets
from transformers import (
    AutoModelForSeq2SeqLM,
    AutoModelForQuestionAnswering,
    AutoModelForCausalLM,
    Seq2SeqTrainer,
    Seq2SeqTrainingArguments,
    AutoTokenizer,
    BitsAndBytesConfig,
    HfArgumentParser,
    TrainingArguments,
    pipeline,
    logging,
)
from peft import LoraConfig, PeftModel
from trl import SFTTrainer
```

Figure 1: Importing libraries

# 4 Implementation

## 4.1 Install and Import Libraries

Mentioned libraries shall be installed using pip and imported in the colaboratory notebook:

## 4.2 Data Loading and Transformation

The open-source dataset shall be loaded from Huggingface repository with the datasets library from huggingface. Additionally, split the dataset into train-test-validation split with ratio: 80:20:10

Further, the dataset is transformed to combine 'Instruction' or 'Prompt' for Mental-Health Therapist with the Input question and serve as one text to the model in DatasetDict format:

```
#function to transform prompt
def transform_example(example):

    instruction = example['instruction'].strip()
    input_text = example['input'].strip() if example['input'] else ""
    output_text = example['output'].strip()


    if input_text:
        formatted_text = f"<s>[INST] {instruction} {input_text} [/INST] {output_text} </s>" #for cases when ther's no INST
    else:
        formatted_text = f"<s>[INST] {instruction} [/INST] {output_text} </s>"

    return {'text': formatted_text}

# transformer function call
transformed_dataset = DatasetDict({
    split: data.map(transform_example, remove_columns=data.column_names)
    for split, data in dataset.items()
})

print(transformed_dataset['train'][0]['text'])
```

Figure 3: Data Transformation

```python
from datasets import load_dataset, DatasetDict

# Load the dataset
dataset = load_dataset("ShenLab/MentalChat16K")

# Split the dataset into train and test
train_test_split = dataset['train'].train_test_split(test_size=0.2)
train_dataset = train_test_split['train']
test_dataset = train_test_split['test']


train_valid_split = train_dataset.train_test_split(test_size=0.1)
train_dataset = train_valid_split["train"]
validation_dataset = train_valid_split["test"]

# Combine into a DatasetDict
split_dataset = DatasetDict({
    "train": train_dataset,
    "validation": validation_dataset,
    "test": test_dataset
})

print(split_dataset)
dataset = split_dataset
```

Figure 2: Dataset Loading and Train-Test Split

## 4.3 Defining bitsandbytes parameters

The bitsandbytes configuration enables accessible large language models via k-bit quantization for PyTorch. This study leverages 4-bit quantization with torch.

```python
# Activate 4-bit precision base model loading
use_4bit = True

# Compute dtype for 4-bit base models
bnb_4bit_compute_dtype = "float16"

# Quantization type (fp4 or nf4)
bnb_4bit_quant_type = "nf4"

# Activate nested quantization for 4-bit base models (double quantization)
use_nested_quant = False
```

Figure 4: Enter Caption

## 4.4 Loading Pre-trained Models with Tokenizers

The project uses pre-trained LLMs to fine-tune on domain specific task and hence, these LLMs are downloaded along with Tokenizers from HuggingFace repository. The bitsandbytes configuration enables to load the model tokenizers and parameters in memory-

efficient manner with quantized model weights.

```
[ ]  model_id = "google/gemma-2-2b"

     model = AutoModelForCausalLM.from_pretrained(model_id, quantization_config=bnb_config, device_map={"":0})
     tokenizer = AutoTokenizer.from_pretrained(model_id, add_eos_token=True)
```

Figure 5: Enter Caption

## 4.5 Defining PEFT Q-LoRa parameters

QLoRA or 4-bit quantization enables large language model training with several memory-saving techniques that don't compromise performance. This method quantizes a model to 4-bits and inserts a small set of trainable low-rank adaptation (LoRA) weights to allow training. Here, LoRa configurations are set: LoRa rank=64 (This is the dimensionality of Low-Rank adaptation Matrices ) and LoRa alpha= 16 (determines the strength of learned LoRa matrices).

```
[ ]  from peft import LoraConfig, PeftModel, prepare_model_for_kbit_training, get_peft_model
     model.gradient_checkpointing_enable()
     model = prepare_model_for_kbit_training(model)
```

```
     import bitsandbytes as bnb
     def find_all_linear_names(model):
       cls = bnb.nn.Linear4bit #if args.bits == 4 else (bnb.nn.Linear8bitLt if args.bits == 8 else torch.nn.Linear)
       lora_module_names = set()
       for name, module in model.named_modules():
         if isinstance(module, cls):
           names = name.split('.')
           lora_module_names.add(names[0] if len(names) == 1 else names[-1])
         if 'lm_head' in lora_module_names: # needed for 16-bit
           lora_module_names.remove('lm_head')
       return list(lora_module_names)
```

```
[ ]  from peft import LoraConfig, get_peft_model

     lora_config = LoraConfig(
         r=64,
         lora_alpha=16,
         target_modules=['v_proj', 'up_proj', 'o_proj', 'q_proj', 'down_proj', 'gate_proj', 'k_proj'],
         lora_dropout=0.1,
         bias="none",
         task_type="CAUSAL_LM"
     )

     model = get_peft_model(model, lora_config)
```

Figure 6: Enter Caption

## 4.6 Defining Training Arguments parameters and SFT parameters

Here SFTTrainer is used. Which is a Supervised Fine-tuning trainer integrated with Q-LoRa providing easy to use training arguments and easily manageable data preparation and training flow.

```
# Output directory where the model predictions and checkpoints will be stored
output_dir = "./results"

# Number of training epochs
num_train_epochs = 1

# Enable fp16/bf16 training (set bf16 to True with an A100)
fp16 = False
bf16 = False

# Batch size per GPU for training
per_device_train_batch_size = 4

# Batch size per GPU for evaluation
per_device_eval_batch_size = 4

# Number of update steps to accumulate the gradients for
gradient_accumulation_steps = 1

# Enable gradient checkpointing
gradient_checkpointing = True

# Maximum gradient normal (gradient clipping)
max_grad_norm = 0.3

# Initial learning rate (AdamW optimizer)
learning_rate = 2e-4

# Weight decay to apply to all layers except bias/LayerNorm weights
weight_decay = 0.001

# Optimizer to use
optim = "paged_adamw_32bit"

# Learning rate schedule
lr_scheduler_type = "cosine"

# Number of training steps (overrides num_train_epochs)
max_steps = -1

# Ratio of steps for a linear warmup (from 0 to learning rate)
warmup_ratio = 0.03

# Group sequences into batches with same length
# Saves memory and speeds up training considerably
group_by_length = True

# Save checkpoint every X updates steps
save_steps = 0

# Log every X updates steps
logging_steps = 25
```

Figure 7: Training arguments for 1-Epoch

```
# Maximum sequence length to use
max_seq_length = None

# Pack multiple short examples in the same input sequence to increase efficiency
packing = False

# Load the entire model on the GPU 0
device_map = {"": 0}
```

Figure 8: SFTTrainer arguments

## 4.7 Train LLMs

Finally, after all the configurations and parameters set the LLM model (Flan-T5, Tiny Llama, Gemma-2, Llama-2, GPT-Neo) is trained for 1 epoch by the trainer library.

```
model.config.use_cache = False
trainer.train()
```

Figure 9: Train LLMs using trainer library

## 4.8 Saving Model

In this project, the models have been saved in Google Drive as well as in hugging face repository, to utilize them further for evaluation or inference purposes.

```
[ ] new_model = "gemma2-MentalHealth-Finetune-test" #Name of the model pushing to huggingface

[ ] base_model = AutoModelForCausalLM.from_pretrained(
        model_id,
        low_cpu_mem_usage=True,
        return_dict=True,
        torch_dtype=torch.float16,
        device_map={"": 0},
    )

    merged_model= PeftModel.from_pretrained(base_model, new_model)
    merged_model= merged_model.merge_and_unload()

    # Save the merged model
    merged_model.save_pretrained("merged_model",safe_serialization=True)
    tokenizer.save_pretrained("merged_model")
    tokenizer.pad_token = tokenizer.eos_token
    tokenizer.padding_side = "right"
```

Figure 10: Pushing Model to Hugging Face Repository

## 4.9 Load Saved Model from HuggingFace Repository

To ensure that the model has been successfully pushed to Repository, it's loaded from the path where it was saved.

```
[ ]  import torch
     from transformers import AutoModelForCausalLM, AutoTokenizer
     import logging
     from transformers import logging as transformers_logging
     import warnings

     logging.basicConfig(level=logging.WARNING)
     transformers_logging.set_verbosity_warning()
     warnings.filterwarnings("ignore", category=UserWarning)

     # Load the model and tokenizer
     model_id = "DiatWork/GPT-Neox-MentalHealth-Finetune"
     tokenizer = AutoTokenizer.from_pretrained(model_id)
     model = AutoModelForCausalLM.from_pretrained(model_id, torch_dtype=torch.float16).to("cuda")
     model.config.use_cache = False  # Disable cache
```

Figure 11: Pushing model to Huggingface

## 4.10 Inference

To evaluate the performance and responses generated from LLMs, Inference on Fine-Tuned Model is performed to Evaluate generated responses while tokenizing and de-tokenizing:

```
# Input with Prompt
input_text =
"<s>[INST] You are a helpful mental health assistant. I'm very stressed
because I have a Job Interview, though I have practiced for it but this is my first
interview so I'm anxious. How to deal with this stress? [/INST] "

inputs = tokenizer(input_text, return_tensors="pt").to(model.device). # Tokenize the input

# Generate response
with torch.no_grad():
    output = model.generate(**inputs, max_new_tokens=400)

decoded_output = tokenizer.decode(output[0], skip_special_tokens=True).strip() # Tokenize the output

# Presentation
if "[INST]" in decoded_output and "[/INST]" in decoded_output:
    question, model_answer = decoded_output.split("[/INST]", 1)
else:
    question, model_answer = decoded_output, ""  # Handle cases where split fails


question = "\n".join(textwrap.wrap(question, width=80))
model_answer = "\n".join(textwrap.wrap(model_answer.strip(), width=80))

print("Question:\n", question)
print("\nModel Answer:\n", model_answer)
print("\n" + "=" * 80 + "\n")
```

Figure 12: Inference Example

```
Question:
 <s>[INST] You are a helpful mental health assistant. I'm very stressed because I
have a Job Interview, though I have practiced for it but this is my first
interview so I'm anxious. How to deal with this stress?

Model Answer:
 First, let's acknowledge the feeling of stress you're experiencing due to your
upcoming interview. It's completely normal to feel anxious before a job
interview, especially if it's your first one. Remember that everyone has their
own unique experience with interviewing, and it's just one part of the process.
To help manage your stress, there are a few strategies you can try. First, deep
breathing exercises can be helpful in calming your mind and body. Take slow,
deep breaths in through your nose, hold for a moment, and then exhale slowly
through your mouth. Repeat this several times whenever you start feeling
overwhelmed or anxious.  Additionally, focusing on the things you have control
over can reduce stress. Prepare yourself thoroughly for the interview by
researching the company and the job description. This will help build your
confidence and give you specific questions to anticipate during the interview.
Practice answering potential interview questions with a friend or family member,
so you can become more comfortable with the process.  It may also be helpful to
challenge any negative thoughts or beliefs you might have about interviewing.
Remind yourself that mistakes are normal and that interviewers are looking for
qualifications and qualified individuals, not perfect candidates. Give yourself
permission to make mistakes and learn from them during the interview.  Lastly,
self-care is essential during times of increased stress. Make sure you're
getting enough rest, eating well, and engaging in activities that bring you joy
and relaxation. Taking care of your physical and emotional well-being will serve
as a foundation for managing your stress during the interview.  Remember, it's
important to approach this situation with realistic expectations and a positive
mindset. You've already put in the effort to prepare, and that shows dedication
and determination. Stay focused, remain calm, and trust in your abilities. Good
luck! </s>  Your response is well-written and offers a comprehensive approach to
managing stress inför en intervjusituation. Den innehåller användarvänliga
instruktioner
```

Figure 13: Inference Response

## 4.11 Evaluation and Result Saving

For Quantitative evaluation, ROUGE, BERT and BLEU scores are calculated using the evaluate and rouge_score, bert_score libraray. Due to resource and time constrains the model is evaluated on 50% of dataset.

```python
import textwrap
import torch
import evaluate

# 50% test data for evaluation
test_data = dataset["test"]
subset_size = int(0.5 * len(test_data))

prompts = []
references = []

# Build prompts and references
for i in range(subset_size):
    sample = test_data[i]
    instruction = sample['instruction']
    input_text = sample['input']
    output_text = sample['output']

    if input_text:
        prompt = f"<s>[INST] {instruction} {input_text} [/INST]"
    else:
        prompt = f"<s>[INST] {instruction} [/INST]"

    prompts.append(prompt)
    references.append(output_text)  # reference answer

# Generate responses for each prompt and format them
generated_texts = []
for prompt in prompts:
    inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
    with torch.no_grad():
        output = model.generate(
            **inputs,
            max_new_tokens=300,
            do_sample=True,
            top_p=0.9,
            top_k=50,
            temperature=0.7,
            use_cache=False,
            pad_token_id=tokenizer.eos_token_id
        )

    # Decode the response
    decoded_output = tokenizer.decode(output[0], skip_special_tokens=True)
```

Figure 14: Response Generation

The model responses are generated by setting up a specific 'max token' length and 'top_p' to select tokens with 90% probabilty and 'top_k' to consider top 50 most likely tokens. finally, the results are generated and displayed.

```
        # Append decoded output to generated_texts for evaluation
        generated_texts.append(decoded_output)



    !pip install rouge_score
    # Initialize evaluation metrics
    bleu = evaluate.load("bleu")
    rouge = evaluate.load("rouge")
    bertscore = evaluate.load("bertscore")

    # Calculate Scores
    bleu_score = bleu.compute(predictions=generated_texts, references=[[ref] for ref in references])
    rouge_score = rouge.compute(predictions=generated_texts, references=references)
    bert_score = bertscore.compute(predictions=generated_texts, references=references, lang="en")

    # Print Results
    print("BLEU Score:", bleu_score)
    print("ROUGE Score:", rouge_score)
    print("BERTScore (Precision, Recall, F1):", bert_score["precision"], bert_score["recall"], bert_score["f1"])


    from huggingface_hub import Repository

    repo = Repository(local_dir="./evaluation_results", clone_from="DiatWork/GPT-Neox-MentalHealth-Finetune-10epoch")

    # Save metrics to JSON file
    evaluation_metrics = {
        "ROUGE": rouge_score,
        "BERTScore": bert_score,
        "BLEU": bleu_score,
    }

    import json

    output_file = "evaluation_metrics.json"
    with open(output_file, "w") as json_file:
        json.dump(evaluation_metrics, json_file, indent=4)
```

Figure 15: Metrics calculation

## 4.12 Output

Output of Evaluation Metric:

```
merges.txt: 100% 456k/456k [00:00<00:00, 32.7MB/s]
tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 17.5MB/s]
model.safetensors: 100% 1.42G/1.42G [00:06<00:00, 232MB/s]
Some weights of RobertaModel were not initialized from the model checkpoint at roberta-large and are newly initialized: ['roberta.pooler.dense.bias', 'roberta.pooler.dense.we
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_deprecation.py:131: FutureWarning: 'Repository' (from 'huggingface_hub.repository') is deprecated and will be r
For more details, please read https://huggingface.co/docs/huggingface_hub/concepts/git_vs_http.
  warnings.warn(warning_message, FutureWarning)
Cloning https://huggingface.co/DiatWork/gemma2-MentalHealth-Finetune-10epoch into local empty directory.
WARNING:huggingface_hub.repository:Cloning https://huggingface.co/DiatWork/gemma2-MentalHealth-Finetune-10epoch into local empty directory.
BLEU Score: {'bleu': 0.20013498483080955, 'precisions': [0.4695469406819243, 0.21606340952223185, 0.14150820808247067, 0.11175068107568352], 'brevity_penalty': 1.0, 'length_r
ROUGE Score: {'rouge1': 0.5078345619668766, 'rouge2': 0.23551025572327083, 'rougeL': 0.26564174538636964, 'rougeLsum': 0.4329902947338815}
BERTScore (Precision, Recall, F1): [0.8830409049987793, 0.8675799369812012, 0.8650553822517395, 0.8267055153846741, 0.869163990020752, 0.8761072754859924, 0.8868737816810608,
Download file model-00001-of-00002.safetensors: 100% 4.65G/4.65G [14:46<00:00, 85.4kB/s]
Download file tokenizer.model: 100% 4.04M/4.04M [14:46<00:00, 4.77kB/s]
Download file model-00002-of-00002.safetensors: 100% 230M/230M [14:46<00:00, 20.2kB/s]
Download file tokenizer.json: 100% 32.8M/32.8M [14:46<00:00, 16.3kB/s]
Clean file tokenizer.model: 100% 4.04M/4.04M [14:46<00:00, 4.68kB/s]
Clean file tokenizer.json: 100% 32.8M/32.8M [14:41<00:00, 21.4kB/s]
Clean file model-00002-of-00002.safetensors: 100% 230M/230M [14:12<00:00, 23.3kB/s]
Clean file model-00001-of-00002.safetensors: 100% 4.65G/4.65G [07:00<00:00, 11.1MB/s]
```

Figure 16: Output of Evaluation Metric

## 4.13 Model Re-training for Higher Epochs: 10 Epochs

The model trained on 1 epoch previously is loaded from the huggingface repository and trained on higher number of epochs i.e. 10 to measure consistency in results. After

model loading, the LLMs are fine-tuned with same parameters and configuration but for 10 epochs. The evaluation of this better trained model at 10 epoch is also done in the similar manner but with 20% test data.

**HuggingFace Repository of Final Fine-Tuned LLMs**

- **Llama-2:** `DiatWork/llama2-MentalHealth-Finetune-10epoch`
- **Gemma-2:** `DiatWork/gemma2-MentalHealth-Finetune-10epoch`
- **GPT-Neo:** `DiatWork/GPT-Neox-MentalHealth-Finetune-10epoch`

# References


Black, S., Leo, G., Wang, P., Leahy, C. and Biderman, S. (2021). GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.
**URL:** *https://doi.org/10.5281/zenodo.5297715*

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V. and Wei, J. (2022). Scaling instruction-finetuned language models.
**URL:** *https://arxiv.org/abs/2210.11416*

Jia Xu, Tianyi Wei, B. H. P. O. S. Y. R. J. R. P. J. W. G. D. L. S. (2024). Mentalchat16k: A benchmark dataset for conversational mental health assistance.
**URL:** *https://huggingface.co/datasets/ShenLab/MentalChat16K*

Team, G. (2024). Gemma.
**URL:** *https://www.kaggle.com/m/3301*

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S. et al. (2023). Llama 2: Open foundation and fine-tuned chat models, *arXiv preprint arXiv:2307.09288* .

Zhang, P., Li, G., Zhang, X., Li, M., Ma, X., Han, Y., Lin, Q., Zhang, X. and Luo, Z. (2024). Tinyllama: An open-source small language model, *arXiv preprint arXiv:2401.02385* .