```python
from unsloth import FastLanguageModel
import torch

class LoRAInferencer:
    """
    Class to load a base model + LoRA adapter and run instruction-based
inference.
    """
    def __init__(self, base_model_path, trained_lora_model_path,
max_seq_length=2048, load_in_4bit=True, dtype=None):
        """
        Initialize the model and tokenizer using the base and LoRA paths.
        """
        self.base_model_path = base_model_path
        self.trained_lora_model_path = trained_lora_model_path
        self.max_seq_length = max_seq_length
        self.load_in_4bit = load_in_4bit
        self.dtype = dtype

        # ✅ Load base model
        self.model, self.tokenizer = FastLanguageModel.from_pretrained(
            model_name=self.base_model_path,
            max_seq_length=self.max_seq_length,
            dtype=self.dtype,
            load_in_4bit=self.load_in_4bit,
        )

        # ✅ Prepare model for inference
        FastLanguageModel.for_inference(self.model)

        # ✅ Now merge in the trained LoRA
        # ✅ Load the LoRA adapter
        self.model.load_adapter(self.trained_lora_model_path)

    def generate_response(self, instruction: str, input_text: str = "",
max_new_tokens: int = 128) -> str:
        """
        Generates a response based on an instruction and optional input.
        """
        # 🧠 Prompt template based on Alpaca format
        alpaca_prompt = """Below is an instruction that describes a task,
paired with an input that provides further context. Write a response that
appropriately completes the request.

        ### Instruction:
        {}

        ### Input:
        {}

        ### Response:
        {}"""

        prompt = alpaca_prompt.format(instruction, input_text, "")
        inputs = self.tokenizer([prompt], return_tensors="pt").to("cuda")

        outputs = self.model.generate(
            **inputs,
            max_new_tokens=max_new_tokens,
            use_cache=True
        )

        # ✅ Decode and clean the output
        decoded_output = self.tokenizer.batch_decode(outputs,
skip_special_tokens=False)[0]
        response = decoded_output.split("### Response:")[-1].strip()
        response = response.split("<eos>")[0].strip()

        return response


# ✅ Usage Example
if __name__ == "__main__":
    base_model_dir =
"/home/gflmltpc/Projects/Gen_AI/LLM_Fine_Tuning/gemma-7b-it-bnb-4bit"
    trained_lora_model_path =
"/home/gflmltpc/Projects/Gen_AI/LLM_Fine_Tuning/trained-gemma-7b-4bit-mode
l"

    inferencer = LoRAInferencer(base_model_dir, trained_lora_model_path)
    while(True):
        user_query = input("you : ")
        if(user_query.lower()=='exit'):
            print("Chatbot session ended.")
            break
        response = inferencer.generate_response(user_query)
        print("\n🧪 Response Only:\n", response)
```