```
# Cell 1: Setup, GPU check, and Login

# 1. Install Dependencies
!pip install -q -U transformers peft accelerate bitsandbytes trl datasets google-generativeai firebase-admin google-auth google-c

# 2. Import and Login
import huggingface_hub
import torch
import json
from datasets import Dataset
import google.generativeai as genai
import firebase_admin
from firebase_admin import credentials, firestore
from google.colab import userdata

print("Logging into Hugging Face...")
huggingface_hub.login()

# 3. Check for GPU
if not torch.cuda.is_available():
    raise SystemError("GPU not available. Please go to Runtime > Change runtime type and select T4 GPU.")
else:
    print("\n✅ GPU is available and ready.")
    !nvidia-smi
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 216.1/216.1 kB 6.5 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is
google-colab 1.0.0 requires google-auth==2.38.0, but you have google-auth 2.40.3 which is incompatible.
gcsfs 2025.3.2 requires fsspec==2025.3.2, but you have fsspec 2025.3.0 which is incompatible.
Logging into Hugging Face...

✅ GPU is available and ready.
Fri Jul  4 16:07:03 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   45C    P8              9W /   70W |       2MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

```
# Cell 2: The Bulletproof Data Parser

def find_and_clean_json_objects(text):
    """
    Scans through a string, finds all valid JSON objects, cleans them,
    and returns a list of clean dictionaries. This is a robust,
    character-level parser.
    """
    cleaned_data = []
    pos = 0
    while pos < len(text):
        # Find the start of the next potential JSON object
        start_brace = text.find('{', pos)
        if start_brace == -1:
            break # No more objects in the string

        # Find the matching end brace
        brace_level = 1
        for i in range(start_brace + 1, len(text)):
            char = text[i]
            if char == '{':
                brace_level += 1
            elif char == '}':
                brace_level -= 1
```

````
            if brace_level == 0:
                end_brace = i
                # We found a complete object, extract it
                potential_json_str = text[start_brace : end_brace + 1]

                try:
                    # Attempt to parse this extracted string
                    data = json.loads(potential_json_str)

                    # Now, clean the 'output' field inside the valid JSON
                    output_str = data.get('output', '')

                    if '```json' in output_str:
                        # Extract content between the fences more robustly
                        parts = output_str.split('```json\n', 1)
                        if len(parts) > 1:
                            output_str = parts[1]

                    if '```' in output_str:
                        parts = output_str.rsplit('\n```', 1)
                        output_str = parts[0]

                    # Final validation of the inner JSON
                    json.loads(output_str)
                    data['output'] = output_str

                    cleaned_data.append(data)

                except Exception as e:
                    # If parsing fails, just ignore this chunk and continue
                    pass

                # Move position to after the object we just processed
                pos = end_brace + 1
                break # Break from inner loop to find the next object
        else:
            # If we go through the whole string and don't find a matching brace, we're done.
            break

    return cleaned_data

# --- EXECUTION ---
filepath = 'training_data.jsonl'
print(f"--- Starting definitive parsing of '{filepath}' ---")

with open(filepath, 'r') as f:
    full_content = f.read()

cleaned_list = find_and_clean_json_objects(full_content)

if not cleaned_list:
    raise ValueError("CRITICAL FAILURE: No valid JSON objects could be extracted from the file.")

print(f"\n✅ Successfully parsed and cleaned {len(cleaned_list)} examples.")

# Create the Hugging Face Dataset
dataset = Dataset.from_list(cleaned_list)

print("\nDataset loaded successfully:")
print(dataset)
````

```
--- Starting definitive parsing of 'training_data.jsonl' ---

✅ Successfully parsed and cleaned 67 examples.

Dataset loaded successfully:
Dataset({
    features: ['instruction', 'input', 'output'],
    num_rows: 67
})
```

```
# Cell 3: Configure, Train, and Upload

from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig, TrainingArguments
from peft import LoraConfig
from trl import SFTTrainer
```

```
# 1. Load Model and Tokenizer
model_id = "meta-llama/Meta-Llama-3-8B-Instruct"
bnb_config = BitsAndBytesConfig(load_in_4bit=True, bnb_4bit_quant_type="nf4", bnb_4bit_compute_dtype=torch.bfloat16)

print("\nLoading base model and tokenizer...")
# Make sure you have requested and been granted access to this model on Hugging Face
model = AutoModelForCausalLM.from_pretrained(model_id, quantization_config=bnb_config, device_map="auto")
tokenizer = AutoTokenizer.from_pretrained(model_id)
tokenizer.pad_token = tokenizer.eos_token
print("✅ Model and tokenizer loaded.")
```

```
Loading base model and tokenizer...
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%                                  654/654 [00:00<00:00, 67.8kB/s]
model.safetensors.index.json: 100%                 23.9k/23.9k [00:00<00:00, 2.62MB/s]
Fetching 4 files: 100%                             4/4 [20:49<00:00, 1249.60s/it]
model-00002-of-00004.safetensors: 100%             5.00G/5.00G [19:34<00:00, 4.12MB/s]
model-00001-of-00004.safetensors: 100%             4.98G/4.98G [20:49<00:00, 12.7MB/s]
model-00004-of-00004.safetensors: 100%             1.17G/1.17G [14:03<00:00, 700kB/s]
model-00003-of-00004.safetensors: 100%             4.92G/4.92G [19:32<00:00, 1.60MB/s]
Loading checkpoint shards: 100%                    4/4 [01:22<00:00, 18.18s/it]
generation_config.json: 100%                       187/187 [00:00<00:00, 15.6kB/s]
tokenizer_config.json: 100%                        51.0k/51.0k [00:00<00:00, 4.65MB/s]
tokenizer.json: 100%                               9.09M/9.09M [00:00<00:00, 26.2MB/s]
special_tokens_map.json: 100%                      73.0/73.0 [00:00<00:00, 8.27kB/s]
✅ Model and tokenizer loaded.
```

```
# 2. Configure LoRA (No changes here)
lora_config = LoraConfig(r=16, lora_alpha=32, lora_dropout=0.05, target_modules=["q_proj", "k_proj", "v_proj", "o_proj"], bias="
```

```
# 3. Define the Formatting Function (No changes here)
def formatting_func(example):
    text_list = []
    for i in range(len(example["instruction"])):
        instruction = example["instruction"][i]
        input_text = example["input"][i]
        output_data = json.loads(example["output"][i])

        output_text = f"Event Summary: {output_data.get('event_summary', 'N/A')}\n"
        output_text += f"Event Type: {output_data.get('event_type', 'N/A')}\n"
        output_text += "First-Order Impacts:\n"
        for impact in output_data.get('first_order_impacts', []):
            output_text += f"- {impact}\n"
        output_text += "Second-Order Hypotheses:\n"
        for h in output_data.get('second_order_hypotheses', []):
            hyp = h.get('hypothesis', 'N/A')
            rsn = h.get('reasoning', 'N/A')
            cnf = h.get('confidence_score', 'N/A')
            output_text += f"- HYPOTHESIS: {hyp} REASONING: {rsn} (CONFIDENCE: {cnf})\n"

        text_list.append(f"<|begin_of_text|><|start_header_id|>system<|end_header_id|>\n\n{instruction}<|eot_id|><|start_header_

    return text_list
```

```
# 4. Set up the Trainer (with memory-saving changes)
training_args = TrainingArguments(
    output_dir="./kairos-llama3-finetune",
    per_device_train_batch_size=1,      # This is already at the minimum
    gradient_accumulation_steps=4,     # Accumulate gradients over 4 steps
    learning_rate=2e-4,
    num_train_epochs=3,
    logging_steps=10,
    save_strategy="epoch",
    fp16=True,
    report_to="none",
    # --- MEMORY SAVING CHANGES ---
    gradient_checkpointing=True,          # ✅ KEY CHANGE: This saves a lot of memory at a small cost of speed.
    optim="paged_adamw_8bit"              # ✅ KEY CHANGE: Use a more memory-efficient optimizer.
)


trainer = SFTTrainer(
    model=model,
    args=training_args,
    train_dataset=dataset,
    peft_config=lora_config,
    formatting_func=formatting_func,
)
```

```
/usr/local/lib/python3.11/dist-packages/peft/mapping_func.py:73: UserWarning: You are trying to modify a model with PEFT for
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/peft/tuners/tuners_utils.py:190: UserWarning: Already found a `peft_config` attribut
  warnings.warn(
Applying formatting function to train dataset:   0%                                   0/67 [00:00<?, ? examples/s]

Applying formatting function to train dataset: 100%                                   67/67 [00:00<00:00, 2976.73 examples/s]

Adding EOS to train dataset: 100%                                    67/67 [00:00<00:00, 2315.60 examples/s]

Tokenizing train dataset: 100%                                    67/67 [00:00<00:00, 592.75 examples/s]

Truncating train dataset: 100%                                    67/67 [00:00<00:00, 5527.29 examples/s]
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if labe
```

```
# 5. Start Training
print("\n--- Starting Final Fine-Tuning Run (Memory Optimized) ---")
trainer.train()
print("✅ Fine-tuning complete!")
```

```

--- Starting Final Fine-Tuning Run (Memory Optimized) ---
`use_cache=True` is incompatible with gradient checkpointing. Setting `use_cache=False`.
                                    [51/51 08:07, Epoch 3/3]

Step  Training Loss
  10       1.453400
  20       0.987900
  30       0.894200
  40       0.825700
  50       0.775900
✅ Fine-tuning complete!
```

```
# 6. Save and Upload to Hub
print("\n--- Saving Model Adapter to Hugging Face Hub ---")
hf_username = huggingface_hub.whoami()['name']
new_model_name = f"{hf_username}/kairos-llama3-finetune"
```

```

--- Saving Model Adapter to Hugging Face Hub ---
```

```
trainer.push_to_hub(new_model_name)
print(f"\n🚀 Process complete! Your model is available at: https://huggingface.co/{new_model_name}")
```

```
No files have been modified since last commit. Skipping to prevent empty commit.
WARNING:huggingface_hub.hf_api:No files have been modified since last commit. Skipping to prevent empty commit.

🚀 Process complete! Your model is available at: https://huggingface.co/SharathReddy/kairos-llama3-finetune
```

Start coding or generate with AI.