```
!nvidia-smi
```

```
Thu May 30 14:57:54 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| 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  NVIDIA L4                      Off | 00000000:00:03.0 Off |                    0 |
| N/A   48C    P8              12W /  72W |      1MiB / 23034MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

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

```
!pip install openai
!pip install gradio

import IPython
import sys
import os

# Run the installation commands
if 'google.colab' in sys.modules:
    print("Running in Google Colab")
    !pip install bitsandbytes accelerate
else:
    print("Not running in Google Colab")
    !pip install transformers accelerate datasets bitsandbytes

import openai


def clean_notebook():
    IPython.display.clear_output(wait=True)
    print("Notebook cleaned.")

# Clean up the notebook
clean_notebook()
```

```
Notebook cleaned.
```

```

openai.api_key = "sk-proj-sQO0S1mGBrbhcgO2bGpDT3BlbkFJpOG6rn9OfMM1SLdq14yO"

os.environ['HF_TOKEN'] ="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"
hf_token               = os.environ['HF_TOKEN']
```

```
# Add information from text file
if 'google.colab' in sys.modules:
  from google.colab import drive
  drive.mount('/content/drive')

  with open("/content/drive/My Drive/Colab Notebooks/MasterAI KMITL Training/stx_history.txt", 'r') as f:
    history = f.read()

  drive.flush_and_unmount()

else:
  print("Not running in Google Colab")
  with open("./stx_history.txt", 'r', errors="ignore") as f:
    history = f.read()

#IPython.display.Markdown(history)

# Define system expertise
system_prompt = """You are a technology blogger whos has more than 15 years \
experience working in IT company. Your answers should be \
short, concise, and clear. If you don't know the answer, say Sorry, I can't \
find a good answer for you.
"""
```

```
Mounted at /content/drive
```

```
del model_typhoon, tokenizer_typhoon
torch.cuda.empty_cache()
```

```
from transformers import AutoModelForCausalLM,AutoTokenizer,BitsAndBytesConfig
import torch

################################################################################
# bitsandbytes parameters
################################################################################

# 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


# Load tokenizer and model with QLoRA configuration
compute_dtype = getattr(torch, bnb_4bit_compute_dtype)

bnb_config = BitsAndBytesConfig(
    load_in_4bit=use_4bit,
    bnb_4bit_quant_type=bnb_4bit_quant_type,
    bnb_4bit_compute_dtype=compute_dtype,
    bnb_4bit_use_double_quant=use_nested_quant,
)


model_typhoon    = AutoModelForCausalLM.from_pretrained("scb10x/llama-3-typhoon-v1.5-8b-instruct", quantization_config=bnb_config,device_map="auto",token=hf_toke
tokenizer_typhoon = AutoTokenizer.from_pretrained("scb10x/llama-3-typhoon-v1.5-8b-instruct")

model_Llama    = AutoModelForCausalLM.from_pretrained("meta-llama/Meta-Llama-3-8B-Instruct", quantization_config=bnb_config,device_map="auto",token=hf_token)
tokenizer_Llama = AutoTokenizer.from_pretrained("meta-llama/Meta-Llama-3-8B-Instruct")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: 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 it as secret in y
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%                                   616/616 [00:00<00:00, 54.0kB/s]
model.safetensors.index.json: 100%                  23.9k/23.9k [00:00<00:00, 1.96MB/s]
Downloading shards: 100%                            4/4 [00:47<00:00, 10.17s/it]
model-00001-of-00004.safetensors: 100%              4.98G/4.98G [00:16<00:00, 372MB/s]
model-00002-of-00004.safetensors: 100%              5.00G/5.00G [00:12<00:00, 429MB/s]
model-00003-of-00004.safetensors: 100%              4.92G/4.92G [00:14<00:00, 419MB/s]
model-00004-of-00004.safetensors: 100%              1.17G/1.17G [00:02<00:00, 411MB/s]
Loading checkpoint shards: 100%                     4/4 [00:08<00:00, 1.85s/it]
generation_config.json: 100%                        194/194 [00:00<00:00, 18.2kB/s]
tokenizer_config.json: 100%                         51.0k/51.0k [00:00<00:00, 4.51MB/s]
tokenizer.json: 100%                                9.09M/9.09M [00:00<00:00, 9.44MB/s]
special_tokens_map.json: 100%                       449/449 [00:00<00:00, 37.4kB/s]
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
config.json: 100%                                   654/654 [00:00<00:00, 57.5kB/s]
model.safetensors.index.json: 100%                  23.9k/23.9k [00:00<00:00, 2.06MB/s]
Downloading shards: 100%                            4/4 [00:43<00:00, 9.51s/it]
model-00001-of-00004.safetensors: 100%              4.98G/4.98G [00:11<00:00, 442MB/s]
model-00002-of-00004.safetensors: 100%              5.00G/5.00G [00:13<00:00, 411MB/s]
model-00003-of-00004.safetensors: 100%              4.92G/4.92G [00:13<00:00, 439MB/s]
model-00004-of-00004.safetensors: 100%              1.17G/1.17G [00:02<00:00, 391MB/s]
Loading checkpoint shards: 100%                     4/4 [00:08<00:00, 1.88s/it]
generation_config.json: 100%                        187/187 [00:00<00:00, 17.0kB/s]
tokenizer_config.json: 100%                         51.0k/51.0k [00:00<00:00, 4.54MB/s]
tokenizer.json: 100%                                9.09M/9.09M [00:00<00:00, 12.9MB/s]
special_tokens_map.json: 100%                       73.0/73.0 [00:00<00:00, 6.66kB/s]
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
```

```
#Typhoon and Llama

def generate_Typhoon_Llama(model, tokenizer,  user_prompt, temperature, max_token=256, top_p=0.9):

    messages = [
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": user_prompt},
    ]

    input_ids = tokenizer.apply_chat_template(messages,add_generation_prompt=True,return_tensors="pt").to(model.device)

    terminators = [
        tokenizer.eos_token_id,
        tokenizer.convert_tokens_to_ids("<|eot_id|>")
    ]

    outputs = model.generate(
        input_ids,
        max_new_tokens=max_token,
        pad_token_id=model.config.eos_token_id,
        eos_token_id=terminators,
        do_sample=True,
        temperature=temperature,
        top_p=top_p,
    )

    response = outputs[0][input_ids.shape[-1]:]
    return tokenizer.decode(response, skip_special_tokens=True)
```

```
# Open AI
def generate_OpenAI(user_prompt, temperature, max_token=256):
    completion = openai.chat.completions.create(
        model='gpt-3.5-turbo',
        messages=[
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": user_prompt},
        ],
        temperature=temperature,
        max_tokens=max_token,
    )
    return completion.choices[0].message.content
```

```
def answer(question, temperature, max_token, model_list):
  prompt = f"""Please answer the following question:

  Question:

  ```{question}```

  Use the following context to find the answer:

  ```{history}```
  """
  #print(prompt)

  result1 = result2 = result3 = "Model is not selected."

  if "OpenAI" in model_list:
    result1 = generate_OpenAI(prompt, temperature, max_token)

  if "Typhoon" in model_list:
    result2 = generate_Typhoon_Llama(model_typhoon, tokenizer_typhoon, prompt, temperature, max_token)

  if "Llama" in model_list:
    result3 = generate_Typhoon_Llama(model_Llama, tokenizer_Llama, prompt, temperature, max_token)

  result = [result1, result2, result3]
  return result
```

```
import gradio as gr

# Example prompts
examples = [
    ["What is Seagate?"],
    ["What are the Seagate products?"],
    ["Current Seagate stock price?"]
]

# Create Gradio interface
interface = gr.Interface(
    fn=answer,
    inputs=[
        gr.Textbox(lines=2, placeholder="Enter your prompt here..."),
        gr.Slider(minimum=0.1, maximum=1.0, step=0.1, value=0.85, label="Temperature"),
        gr.Slider(minimum=0.1, maximum=512, step=1, value=256, label="token"),
        gr.CheckboxGroup(["OpenAI", "Typhoon", "Llama"], label="Model selections", value=["OpenAI", "Typhoon", "Llama"])
    ],
    outputs=[gr.Textbox(label="OpenAI result"),gr.Textbox(label="Typhoon result"),gr.Textbox(label="Llama result")],
    title="Q&A about Seagate",
    description="Ask anything that you want to know about Seagate",
    examples=examples
)

# Launch the interface
interface.launch(share=True)
```

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://c7fc2994fb4c747855.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https
```

Q&A about Seagate

Ask anything that you want to know about Seagate

question

What is Seagate?

Temperature 0.85

token 256

Model selections

OpenAI Typhoon Llama

Clear Submit

OpenAI result

Seagate is an American data storage company that specializes in producing hard disk drives (HDDs), hybrid drives, and solid-state drives (SSDs). It was founded in 1979 as Shugart Technology and has since become a major player in the computer storage industry

Typhoon result

Seagate Technology Holdings plc is an American data storage company. It was initially incorporated in 1978 as Shugart Technology and commenced business in 1979. Seagate developed the first 5.25-inch hard disk drive in 1980. They were a major supplier in the microcomputer market during the 1980s, especially after the introduction of the IBM XT in 1983. Much of their growth has come

Llama result

Seagate is an American data storage company that was incorporated in 1978. It is one of the largest and most well-known hard drive