!pip install -q -U trl transformers==4.28.0 accelerate git+https://github.com/huggingface/peft.git

!pip install -q datasets bitsandbytes einops wandb

!pip install gdown -q

1. It updates or installs the following packages:
   * **trl** (transformer-based reinforcement learning library)
   * **transformers** version 4.28.0 (Hugging Face's Transformers library)
   * **accelerate** (for faster model training)
   * **peft** from a specific GitHub repository (a tool for pretraining transformers efficiently)
2. It installs additional packages:
   * **datasets** (a library for managing and working with datasets)
   * **bitsandbytes** (possibly a custom package)
   * **einops** (a library for tensor operations)
   * **wandb** (Weights and Biases, a platform for machine learning experiment tracking)
3. Finally, it installs the **gdown** package, which is used for downloading files from Google Drive.

!mkdir outputs

!gdown https://drive.google.com/file/d/19TRNTEYyMBcA9JuAOPW8cogVah2CJzpq/view?usp=sharingg -O "outputs/adapter\_model.bin"

1. It creates a directory named "outputs" using the **mkdir** command. This directory will be used to store the output files.
2. It uses the **gdown** command to download a file from a Google Drive link and save it as "adapter\_model.bin" in the "outputs" directory.

import torch

from transformers import AutoModelForCausalLM, AutoTokenizer, BitsAndBytesConfig, AutoTokenizer

model\_name = "meta-llama/Llama-2-13b-hf"

bnb\_config = BitsAndBytesConfig(

    load\_in\_4bit=True,

    bnb\_4bit\_quant\_type="nf4",

    bnb\_4bit\_compute\_dtype=torch.float16,

)

model = AutoModelForCausalLM.from\_pretrained(

    model\_name,

    quantization\_config=bnb\_config,

    trust\_remote\_code=True,

    use\_auth\_token="hf\_dVZJXVJpHqXWJdxbbtKLKYRzxsDLDnHsxF"

)

model.config.use\_cache = False

This code sets up a Python environment for working with a pre-trained language model from the Hugging Face Transformers library, with a focus on efficient inference.

tokenizer = AutoTokenizer.from\_pretrained(model\_name, trust\_remote\_code=True, use\_auth\_token = f"{huggingface\_auth\_token}")

tokenizer.pad\_token = tokenizer.eos\_token

This code initializes a tokenizer for a pre-trained language model and configures it for specific behaviors:

1. It loads a tokenizer associated with the pre-trained model specified by **model\_name**, possibly accessing it remotely using authentication provided by **huggingface\_auth\_token**.
2. It sets the **pad\_token** of the tokenizer to be the same as the **eos\_token** (end-of-sequence token). This means that when sequences are padded for batching, the padding will be applied using the **eos\_token**.

from peft import LoraConfig, get\_peft\_model

This statement import the LoraConfig and get\_peft\_model from peft package

These imported elements suggest that the code may be setting up or configuring a model or algorithm for local representation adaptation or fine-tuning. However, the specific usage and implementation of these elements would require additional code context.

lora\_config = LoraConfig.from\_pretrained('outputs/')

model = get\_peft\_model(model, lora\_config)

this code sets up a model for local representation adaptation using LORA and configures it based on a pre-trained LORA configuration loaded from the 'outputs/' directory. The resulting **model** is tailored for the specific adaptation task.

device = "cuda:0"

# Tokenize the input text

inputs = tokenizer(input, return\_tensors="pt").to(device)

# Remove the 'token\_type\_ids' key from the inputs dictionary

inputs.pop('token\_type\_ids',None)

print(inputs)

max\_len = len(inputs) + 100

print(f"Maximum length : {max\_len}")

# Generate text

outputs = model.generate(\*\*inputs, max\_length=100)  # Note: 'max\_new\_tokens' should be 'max\_length'

print(tokenizer.decode(outputs[0], skip\_special\_tokens=True))

This code snippet performs several operations related to tokenization, text generation, and handling device placement. this code tokenizes input text, prepares it for text generation on the specified device (GPU), generates text using the model, and prints the generated text with a specified maximum length.