Parameter Efficient Fine-tuning of the Gemma model with QLORA

Step 1: Install All the Required Packages

```
!pip install peft datasets transformers trl accelerate bitsandbytes -q
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 190.9/190.9 kB 3.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 510.5/510.5 kB 5.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 155.3/155.3 kB 6.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 280.0/280.0 kB 8.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 102.2/102.2 MB 6.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 9.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 8.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.8/79.8 kB 9.8 MB/s eta 0:00:00
```

Step 2: Import All the Required Libraries

```
import torch
import json
from transformers import AutoModelForCausalLM, AutoTokenizer, BitsAndBytesConfig, TrainingArguments
from peft import LoraConfig, PeftModelForCausalLM
from trl import SFTTrainer
from datasets import Dataset
from huggingface_hub import notebook_login
```

Step 3: HuggingFace loging with token

```
notebook_login()
```

```
Token is valid (permission: read).
Your token has been saved in your configured git credential helpers (store).
Your token has been saved to /root/.cache/huggingface/token
Login successful
```

Step 4: Data preperation

```
f = open(r"/content/english_python_data.txt", "r", encoding="utf8")
file_lines = f.readlines()
```

```
file_lines[:7]
```

```
['# write a python program to add two numbers \n',
 'num1 = 1.5\n',
 'num2 = 6.3\n',
 'sum = num1 + num2\n',
 "print(f'Sum: {sum}')\n",
 '\n',
 '\n']
```

```
dps = []
dp = None
for line in file_lines:
    if line[0] == "#":
        if dp:
            dp['solution'] = ''.join(dp['solution'])
            dps.append(dp)
        dp = {"question": None, "solution": []}
        dp['question'] = line[1:]
    else:
        dp["solution"].append(line)
```

```
dps
```

```
<= j:\n            alist[i], alist[j] = alist[j], alist[i]\n        else:\n            alist[start], alist[j] = alist[j], alist[start]\n            return j\n \nalist = [2, 3,
5, 6, 4, 5]\nquicksort(alist, 0, len(alist))\nprint('Sorted list: ', end='')\nprint(alist)\n\n"},
 {'question': ' Write a python program to Implement Heapsort and print the sorted list for the below list\n',
  'solution': "\ndef heapsort(alist):\n    build_max_heap(alist)\n    for i in range(len(alist) - 1, 0, -1):\n        alist[0], alist[i] = alist[i], alist[0]\n
max_heapify(alist, index=0, size=i)\n \ndef parent(i):\n    return (i - 1)//2\n \ndef left(i):\n    return 2*i + 1\n \ndef right(i):\n    return 2*i + 2\n \ndef
build_max_heap(alist):\n    length = len(alist)\n    start = parent(length - 1)\n    while start >= 0:\n        max_heapify(alist, index=start, size=length)\n        start = start
- 1\n \ndef max_heapify(alist, index, size):\n    l = left(index)\n    r = right(index)\n    if (l < size and alist[l] > alist[index]):\n        largest = l\n    else:\n
largest = index\n    if (r < size and alist[r] > alist[largest]):\n        largest = r\n    if (largest != index):\n        alist[largest], alist[index] = alist[index],
alist[largest]\n        max_heapify(alist, largest, size)\n \n \nalist = [2, 3, 5, 6, 4, 5]\nheapsort(alist)\nprint('Sorted list: ', end='')\nprint(alist)\n\n"},
 {'question': ' Write a python program to Implement Counting sort and print the sorted list for the below list\n',
  'solution': "\ndef counting_sort(alist, largest):\n    c = [0]*(largest + 1)\n    for i in range(len(alist)):\n        c[alist[i]] = c[alist[i]] + 1\n \n    c[0] = c[0] - 1 \n
for i in range(1, largest + 1):\n        c[i] = c[i] + c[i - 1]\n \n    result = [None]*len(alist)\n \n    for x in reversed(alist):\n        result[c[x]] = x\n        c[x] = c[x]
- 1\n \n    return result\n \n \nalist = [2, 3, 5, 6, 4, 5]\nk = max(alist)\nsorted_list = counting_sort(alist, k)\nprint('Sorted list: ', end='')\nprint(sorted_list)\n\n"},
 {'question': ' Write a python program to Implement Radix sort and print the sorted list for the below list\n',
  'solution': "\ndef radix_sort(alist, base=10):\n    if alist == []:\n        return\n \n    def key_factory(digit, base):\n        def key(alist, index):\n            return
((alist[index]//(base**digit)) % base)\n        return key\n    largest = max(alist)\n    exp = 0\n    while base**exp <= largest:\n        alist = counting_sort(alist, base - 1,
key_factory(exp, base))\n        exp = exp + 1\n    return alist\n \ndef counting_sort(alist, largest, key):\n    c = [0]*(largest + 1)\n    for i in range(len(alist)):\n
c[key(alist, i)] = c[key(alist, i)] + 1\n \n    c[0] = c[0] - 1\n    for i in range(1, largest + 1):\n        c[i] = c[i] + c[i - 1]\n \n    result = [None]*len(alist)\n    for i
in range(len(alist) - 1, -1, -1):\n        result[c[key(alist, i)]] = alist[i]\n        c[key(alist, i)] = c[key(alist, i)] - 1\n \n    return result\n \nalist = [2, 3, 5, 6, 4,
5]\nsorted_list = radix_sort(alist)\nprint('Sorted list: ', end='')\nprint(sorted_list)\n\n"},
 {'question': ' Write a python program to Implement Bucket sort and print the sorted list for the below list\n',
  'solution': "\ndef bucket_sort(alist):\n    largest = max(alist)\n    length = len(alist)\n    size = largest/length\n \n    buckets = [[] for _ in range(length)]\n    for i in
range(length):\n        j = int(alist[i]/size)\n        if j != length:\n            buckets[j].append(alist[i])\n        else:\n            buckets[length - 1].append(alist[i])\n
\n    for i in range(length):\n        insertion_sort(buckets[i])\n \n    result = []\n    for i in range(length):\n        result = result + buckets[i]\n \n    return result\n
\ndef insertion_sort(alist):\n    for i in range(1, len(alist)):\n        temp = alist[i]\n        j = i - 1\n        while (j >= 0 and temp < alist[j]):\n            alist[j + 1]
= alist[j]\n            j = j - 1\n        alist[j + 1] = temp\n \n \nalist = [2, 3, 5, 6, 4, 5]\nsorted_list = bucket_sort(alist)\nprint('Sorted list: ',
end='')\nprint(sorted_list)\n\n"},
 {'question': ' Write a python program to Implement Gnome sort and print the sorted list for the below list\n',
  'solution': "\ndef gnome_sort(alist):\n    for pos in range(1, len(alist)):\n        while (pos != 0 and alist[pos] < alist[pos - 1]):\n            alist[pos], alist[pos - 1] =
```


```
data_python_code = []
for dp in dps:
    data_python_code.append(f"Instruction:\n{dp['question'][1:]}\n\nResponse:\n{dp['solution']}")
```

```
len(data_python_code)
```

```
4957
```

```
data_python_code
```

```
entered number is = %d  %reverse)  \n\n ,
 'Instruction:\nwrite a python program to take year as input and check if it is a leap year or not\n\n\nResponse:\n\nyear = int(input("Enter a year: "))  \nif (year % 4) == 0:  \n
if (year % 100) == 0:  \n       if (year % 400) == 0:  \n           print(f"{year} is a leap year")  \n       else:  \n           print(f"{year} is not a leap year")  \n   else:
\n       print(f"{year} is a leap year")  \nelse:  \n   print(f"{year} is not a leap year")\n   \n',
 'Instruction:\nwrite a python program to input a number to test and print if it is a prime number\n\n\nResponse:\n\nnum = int(input("Enter number :"))\nlim = int(num/2) + 1\nfor
i in range(2,lim):\n    rem = num % i\n    if rem == 0 :\n        print(num,"is not a prime number")\n        break\nelse:\n    print(num,"is a prime number")\n    \n',
 'Instruction:\nwrite a python program to input a string from user and convert input string into all upper case and print the result\n\n\nResponse:\nstring = input("Please Enter
your Own String : ")\n\nstring1 = string.upper()\n \nprint("\\nOriginal String in Lowercase  =  ", string)\nprint("The Given String in Uppercase =  ", string1)\n\n',
 'Instruction:\nwrite a python program to input a string from user and count vowels in a string and print the output\n\n\nResponse:\n\nstr1 = input("Please Enter Your Own String :
")\nvowels = 0\n \nfor i in str1:\n    if(i == \'a\' or i == \'e\' or i == \'i\' or i == \'o\' or i == \'u\' or i == \'A\'\n       or i == \'E\' or i == \'I\' or i == \'O\' or i
== \'U\'):\n        vowels = vowels + 1\n \nprint("Total Number of Vowels in this String = ", vowels)\n\n',
 'Instruction:\nwrite a python program to input a Number N from user and print Odd Numbers from 1 to N\n\n\nResponse:\n\nmaximum = int(input(" Please Enter any Maximum Value :
"))\n\nfor number in range(1, maximum + 1):\n    if(number % 2 != 0):\n        print("{0}".format(number))\n        \n',
 'Instruction:\nwrite a python program to input a Number N from user and print Even Numbers from 1 to N\n\n\nResponse:\n\nmaximum = int(input(" Please Enter the Maximum Value :
"))\n\nfor number in range(1, maximum+1):\n    if(number % 2 == 0):\n        print("{0}".format(number))\n        \n',
 'Instruction:\nwrite a python program to input two numbers from user and add two Numbers and print the result\n\n\nResponse:\n\nnumber1 = input(" Please Enter the First Number:
")\nnumber2 = input(" Please Enter the second number: ")\n\nsum = float(number1) + float(number2)\nprint(\'The sum of {0} and {1} is {2}\'.format(number1, number2, sum))\n\n',
 'Instruction:\nwrite a python program that takes two integers as input and check if the first number is divisible by other\n\n\nResponse:\n\nnum1 = int(input("Enter first number
:"))\nnum2 = int(input("Enter second number :"))\nremainder  = num1 % num2\nif remainder == 0:\n    print(num1 ," is divisible by ",num2)\nelse :\n    print(num1 ," is not
divisible by ",num2)\n    \n',
 'Instruction:\nwrite a python program to print the table of input integer\n\n\nResponse:\n\nnum = int(input("Please enter a number "))\nfor a in range(1,11):\n    print(num ,
\'x\' , a , \'=\' ,num*a)\n    \n',
 'Instruction:\nwrite a python program to print the factorial of number\n\n\nResponse:\nnum = int(input("Please enter a number "))\nfact = 1\na = 1\nwhile a <= num :\n    fact *=
a\n    a += 1\nprint("The factorial of ",num, " is ",fact)\n\n',
 'Instruction:\nwrite a python program which takes 3 numbers as input and to print largest of three numbers using elif statement\n\n\nResponse:\n\na = float(input("Please Enter
the First value: "))\nb = float(input("Please Enter the First value: "))\nc = float(input("Please Enter the First value: "))\n\nif (a > b and a > c):\n          print("{0} is
Greater Than both {1} and {2}". format(a, b, c))\nelif (b > a and b > c):\n          print("{0} is Greater Than both {1} and {2}". format(b, a, c))\nelif (c > a and c > b):\n
print("{0} is Greater Than both {1} and {2}". format(c, a, b))\nelse:\n          print("Either any two values or all the three values are equal")\n          \n',
 'Instruction:\nwrite a python program which takes input a number N and print first N elements of fibonacci series\n\n\nResponse:\n\nN = int(input("Please enter a number
"))\nfirst = 0\nsecond = 1\nprint(first)\nprint(second)\nfor a in range(1,N-1):\n    third = first + second\n    print(third)\n    first,second = second , third\n    \n',
 'Instruction:\nwrite a python program to print the divisors of a integer\n\n\nResponse:\nnum = int(input("Please enter a integer "))\nmid = int(num / 2)\nprint("The divisiors of
",num," are :" )\nfor a in range(2,mid + 1):\n    if num % a == 0:\n        print(a, end = \' \')\nelse :\n    print()\n    print("-End-")\n    \n',
 'Instruction:\nwrite a python program to find the average of list of numbers provided as input by user\n\n\nResponse:\nn=int(input("Enter the number of elements to be inserted:
"))\na=[]\nfor i in range(0,n):\n    elem=int(input("Enter element: "))\n    a.append(elem)\navg=sum(a)/n\nprint("Average of elements in the list",round(avg,2))\n\n',
 'Instruction:\nwrite a python program which takes an integer N as input and add the odd numbers up to N and print the result\n\n\nResponse:\n\nN = int(input("Enter Number :
"))\nsum = 0\ni = 1\nwhile i <= N:\n    sum = sum + i\n    i = i + 2\nprint(sum)\n\n',
 'Instruction:\nwrite a python function which takes input a string and returns whether is is a palindrome or not\n\n\nResponse:\n\ndef isPalindrome(s):\n    return s == s[::-1]\n
\n',
```

```
from datasets import Dataset
```

```
dataset ={'text': data_python_code}
```

```
dataset["text"]
```

```
http://checkip.dyndns.org \n    with urllib.urlopen(url) as u:\n        s = str(u.read())\n        ip = re.findall(r'\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}', s)\n        print('IP
Address: ", ip[0])\n        return ip[0]\n\n\n',
 'Instruction:\nwrite a python function for some weird hypnosis text.\n\n\nResponse:\ndef weird():\n    import random\n\n    def getlength(script):\n        return
sum((i[\'length\'] for i in script))\n\n    def truncate(target_length, script):\n        if getlength(script) > target_length:\n            script = sorted(script, key=lambda k:
(k[\'priority\'], -k[\'length\']))[:-1]\n            return truncate(target_length, script)\n        return sorted(script, key=lambda k: k[\'index\'])\n\n    def
as_text(script):\n        return "\\n".join([i[\'text\'] for i in script])\n\n    priorities_and_sentences = [\n        (1, "...now... sitting comfortably in the chair"),\n
(2, "...with your feet still flat on the ground"),\n        (3, "...back straight and head up right"),\n        (2, "...make these adjustments now if you need to"),\n        (3,
"... pause..............................."),\n        (1, "...your eyes ...still ...comfortably closed"),\n        (2, "...nice and relaxed...comfortable and relaxed..."),\n
(3, "... pause....."),\n        (1, "...now...I want you to notice...how heavy your head is starting to feel..."),\n        (1, "how heavy your
head feels..."),\n        (3, "... pause....."),\n        (2, "really noticing the weight... of your head..."),\n        (3,\n         "and how
much more ...comfortable...it will feel when you let your neck relaxes ...and your head begins to fall forward ...into a much more comfortable"),\n    ]\n\n    scriptlist =
[{\'priority\': j[0], \'text\': j[1], \'length\': len(j[1]), \'index\': i} for i, j in\n                    enumerate(priorities_and_sentences)]\n\n    print(as_text(truncate(500,
scriptlist)))\n    print(as_text(truncate(300, scriptlist)))\n    print(as_text(truncate(200, scriptlist)))\n\n\n',
 'Instruction:\nwrite a python function for dice roll asking user for input to continue and randomly give an output.\n\n\nResponse:\ndef dice():\n    import random\n    min = 1\n
max = 6\n    roll_again = \'y\'\n\n    while roll_again == "yes" or roll_again == "y":\n        print("Rolling the dice...")\n        print(random.randint(min, max))\n
roll_again = input("Roll the dices again?")\n\n\nfrom cryptography.fernet import Fernet\n\n\n',
 'Instruction:\nwrite a python program to Encrypt and Decrypt features within \'Secure\' class with key generation, using cryptography module\n\n\nResponse:\nclass Secure:\n
def __init__(self):\n        """\n           Generates a key and save it into a file\n        """\n        key = Fernet.generate_key()\n        with open("secret.key", "wb") as
key_file:\n            key_file.write(key)\n\n    @staticmethod\n    def load_key():\n        """\n        Load the previously generated key\n        """\n        return
open("secret.key", "rb").read()\n\n    def encrypt_message(self, message):\n        """\n        Encrypts a message\n        """\n        key = self.load_key()\n
encoded_message = message.encode()\n        f = Fernet(key)\n        encrypted_message = f.encrypt(encoded_message)\n        print("\\nMessage has been encrypted: ",
encrypted_message)\n        return encrypted_message\n\n    def decrypt_message(self, encrypted_message):\n        """\n        Decrypts an encrypted message\n        """\n
key = self.load_key()\n        f = Fernet(key)\n        decrypted_message = f.decrypt(encrypted_message)\n        print("\\nDecrypted message:", decrypted_message.decode())\n\n\ns
= Secure()\nencrypted = s.encrypt_message("My deepest secret!")\ns.decrypt_message(encrypted)\n\n\n',
 'Instruction:\nwrite a python function to generate SHA256 for given text\n\n\nResponse:\ndef get_sha256(text):\n    import hashlib\n    return
hashlib.sha256(text).hexdigest()\n\n\n',
 'Instruction:\nwrite a python function to check if SHA256 hashed value is valid for given data or not\n\n\nResponse:\ndef check_sha256_hash(hashed, data):\n    import hashlib\n
return True if hashed == hashlib.sha256(data.encode()).hexdigest() else False\n\n\n',
 'Instruction:\nwrite a python function to get HTML code for a given URL\n\n\nResponse:\ndef get_html(url="http://www.python.org"):\n    import urllib.request\n\n    fp =
urllib.request.urlopen(url)\n    mybytes = fp.read()\n    mystr = mybytes.decode("utf8")\n    fp.close()\n    print(mystr)\n\n\n',
 'Instruction:\nwrite a python function to get Bitcoin prices after every given \'interval\' seconds\n\n\nResponse:\ndef get_btc_price(interval=5):\n    import requests\n
import json\n    from time import sleep\n\n    def getBitcoinPrice():\n        URL = "https://www.bitstamp.net/api/ticker/"\n        try:\n            r = requests.get(URL)\n
priceFloat = float(json.loads(r.text)["last"])\n            return priceFloat\n        except requests.ConnectionError:\n            print("Error querying Bitstamp API")\n\n
while True:\n        print("Bitstamp last price: US $ " + str(getBitcoinPrice()) + "/BTC")\n        sleep(interval)\n\n\n',
 "Instruction:\nwrite a python function to get stock prices for a company from 2015 to 2020-12\n\n\nResponse:\ndef get_stock_prices(tickerSymbol='TSLA'):\n    import yfinance as
yf\n\n    # get data on this ticker\n    tickerData = yf.Ticker(tickerSymbol)\n\n    # get the historical prices for this ticker\n    tickerDf = tickerData.history(period='1d',
start='2015-1-1', end='2020-12-20')\n\n    # see your data\n    print(tickerDf)\n\n\n",
 "Instruction:\nwrite a python function to get 10 best Artists playing on Apple iTunes\n\n\nResponse:\ndef get_artists():\n    import requests\n    url =
'https://itunes.apple.com/us/rss/topsongs/limit=10/json'\n    response = requests.get(url)\n    data = response.json()\n    for artist_dict in data['feed']['entry']:\n
artist_name = artist_dict['im:artist']['label']\n        print(artist_name)\n\n\n",
 "Instruction:\nwrite a python function to get prominent words from user test corpus using TFIDF vectorizer\n\n\nResponse:\ndef get_words(corpus, new_doc, top=2):\n    import
numpy as np\n    from sklearn.feature_extraction.text import TfidfVectorizer\n\n    tfidf = TfidfVectorizer(stop_words='english')\n    if not corpus:\n        corpus = [\n
'I would like to check this document',\n            'How about one more document',\n            'Aim is to capture the key words from the corpus',\n            'frequency of words
in a document is called term frequency'\n        ]\n\n    X = tfidf.fit_transform(corpus)\n    feature_names = np.array(tfidf.get_feature_names())\n\n    if not new_doc:\n
```

```
len(dataset["text"])
```

```
4957
```

```
dataset = Dataset.from_dict(dataset)
```

```
dataset
```

```
Dataset({
    features: ['text'],
    num_rows: 4957
})
```

```
dataset_train = dataset.select(range(1000))
```

```
dataset_val = dataset.select(range(1000, 1200))
```

```
len(dataset_train), len(dataset_val)
```

```
(1000, 200)
```

```
dataset_train
```

```
Dataset({
    features: ['text'],
    num_rows: 1000
})
```

```
dataset_train["text"][0]
```

```
'Instruction:\nwrite a python program to add two numbers \n\n\nResponse:\nnum1 = 1.5\nnum2 = 6.3\nsum = num1 + num2\nprint(f'Sum: {sum}')\n\n\n'
```

```
dataset_val
```

```
Dataset({
    features: ['text'],
    num_rows: 200
})
```

```
dataset_val["text"][0]
```

```
'Instruction:\n write a program to get numbers = 1,3,11,42,12,4001\n\n\nResponse:\nfrom collections import Iterable\nhighestnumber = -999\nfor i in numbers:\n  if i > highestnumbe
r:\n    highestnumber = i\nprint(numbers.index(highestnumber))\n\n'
```

Step 4: Load Model - BitsAndBytesConfig int-4 config

```
bnb_config = BitsAndBytesConfig(
  load_in_4bit=True,
  bnb_4bit_quant_type="nf4",
  bnb_4bit_compute_dtype=torch.float16,
)
tokenizer = AutoTokenizer.from_pretrained("google/gemma-2b-it")
tokenizer.padding_side = 'right'
model = AutoModelForCausalLM.from_pretrained(
    "google/gemma-2b-it",
    device_map="auto",
    quantization_config=bnb_config
)
```

```
tokenizer_config.json: 100%  2.16k/2.16k [00:00<00:00, 91.0kB/s]
tokenizer.model: 100%  4.24M/4.24M [00:00<00:00, 51.3MB/s]
tokenizer.json: 100%  17.5M/17.5M [00:00<00:00, 141MB/s]
special_tokens_map.json: 100%  888/888 [00:00<00:00, 46.0kB/s]
config.json: 100%  627/627 [00:00<00:00, 37.7kB/s]
model.safetensors.index.json: 100%  13.5k/13.5k [00:00<00:00, 703kB/s]
Downloading shards: 100%  2/2 [00:39<00:00, 16.49s/it]
model-00001-of-00002.safetensors: 100%  4.95G/4.95G [00:39<00:00, 246MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:00<00:00, 287MB/s]
Loading checkpoint shards: 100%  2/2 [00:23<00:00, 9.82s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 9.43kB/s]
```

Step 5: Lora congif

```
peft_config = LoraConfig(
    lora_alpha = 16,
    lora_dropout=0.1,
    r=16,
    task_type='CAUSAL_LM'
)
```

Step 6: Initialize the Supervised Fine-tuning Trainer

```
training_arguments = TrainingArguments(
    output_dir = "/PATH_TO_TRAINING_OUTPUT",
    evaluation_strategy="steps",
    logging_strategy="steps",
    lr_scheduler_type="constant",
    logging_steps=20,
    eval_steps=20,
    save_steps=20,
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    gradient_accumulation_steps=16,
    eval_accumulation_steps=16,
    num_train_epochs=5,
    fp16=True,
    group_by_length = True,
    optim="paged_adamw_32bit",
    max_steps = 100
```

```
    max_steps = 100
)
trainer = SFTTrainer(
    model,
    tokenizer=tokenizer,
    train_dataset=dataset_train,
    eval_dataset=dataset_val,
    dataset_text_field='text',
    peft_config=peft_config,
    neftune_noise_alpha=5,
    max_seq_length=500,
    args = training_arguments
)
trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/trl/trainer/sft_trainer.py:236: UserWarning: You passed a `neftune_noise_alpha` argument to the SFTTrainer, the value you passed will override
  warnings.warn(
```

Map: 100% 1000/1000 [00:00<00:00, 2749.33 examples/s]

Map: 100% 200/200 [00:00<00:00, 1960.50 examples/s]

[100/100 11:31, Epoch 3/4]

Step	Training Loss	Validation Loss
20	3.916500	3.331199
40	3.456500	2.683542
60	2.712800	2.104791
80	2.112900	1.780471
100	1.851800	1.650420

```
TrainOutput(global_step=100, training_loss=2.8101186752319336, metrics={'train_runtime': 700.7346, 'train_samples_per_second': 4.567, 'train_steps_per_second': 0.143, 'total_flos':
3170575978831872.0, 'train_loss': 2.8101186752319336, 'epoch': 3.2})
```

```
trainer.model.save_pretrained("/SAVE_DIR")
```

```
finetuned_model = PeftModelForCausalLM.from_pretrained(model=model, model_id="/SAVE_DIR")
```

```
messages = [
    {
        'role':'user',
        'content':'Write a python function that returns the sum of n natural numbers?',
    }
]



input_ids = tokenizer.apply_chat_template(messages, add_generation_prompt=True, tokenize=True, return_tensors="pt").to("cuda")

# print(input_ids)
outputs_finetuned = finetuned_model.generate(input_ids=input_ids, max_new_tokens=1024, do_sample=False)


print("finetuned: " + tokenizer.decode(outputs_finetuned[0]).split('<start_of_turn>model\n')[-1])
```

finetuned: ```python
def sum_natural_numbers(n):
    """
    Returns the sum of n natural numbers.

    Args:
        n: The number of natural numbers to sum.

    Returns:
        int: The sum of n natural numbers.
    """

    # Initialize the sum to 0.
    sum = 0

    # Iterate over the numbers from 1 to n.
    for i in range(1, n + 1):
        # Add the current number to the sum.
        sum += i

    # Return the sum.
    return sum

if __name__ == "__main__":
    # Get the number of natural numbers to sum from the user.
    n = int(input("Enter the number of natural numbers to sum: "))

    # Calculate and print the sum.
    sum = sum_natural_numbers(n)
    print(f"The sum of {n} natural numbers is {sum}")
```

**Example Usage:**

```
Enter the number of natural numbers to sum: 10
The sum of 10 natural numbers is 55
```
```

```
**Notes:**

* The function assumes that `n` is a positive integer.
* The time complexity of this function is O(n), where n is the number of natural numbers to sum.
* The function can be modified to handle negative values of `n` by using a different logic.<eos>
```

```
messages = [
    {
        'role':'user',
        'content':'Write a Program to implement validation of a Password',
    }
]
```

```
input_ids = tokenizer.apply_chat_template(messages, add_generation_prompt=True, tokenize=True, return_tensors="pt").to("cuda")

# print(input_ids)
outputs_finetuned = finetuned_model.generate(input_ids=input_ids, max_new_tokens=1024, do_sample=False)
```

```
print("finetuned: " + tokenizer.decode(outputs_finetuned[0]).split('<start_of_turn>model\n')[-1])
```

```
        # Check if the password meets the minimum length requirement.
        if len(password) < 8:
            return False

        # Check if the password contains a mix of upper and lower case letters.
        if not re.search(r'[A-z]', password):
            return False

        # Check if the password contains a number.
        if not re.search(r'\d', password):
            return False

        # Check if the password contains a special character.
        if not re.search(r'!@#$%^&*_~', password):
            return False

        # Check if the password is the same as the original.
        if password == password.lower():
            return False
```

````
if __name__ == "__main__":
    main()
```

**How to use the program:**

1. Run the program.
2. Enter a password when prompted.
3. The program will validate the password and print a message.

**Example output:**

```
Enter a password: P@ssw0rd
Password is valid.
```

**Notes:**

* The password policy in the program is defined by the `password` variable.
* You can modify the password policy by changing the regular expressions used in the `validate_password()` function.
* The program assumes that the password is a string. If you are using a different data type, you can use the `isinstance()` function to check the type of the password.<eos>
````
````