# Folder Structure

GPT2-Finetuning-Code/

README.md

install\_venv.sh

requirements.txt

Research/

1.Base.ipynb

1.Calling base Model.ipynb

# File Paths and Contents

## /mnt/d/Projects/LLM\_finetuneing\_and\_Compression/Report/1.Completed\_Imdb\_Finetuning/GPT2-Finetuning-Code/README.md

# GPT2-Finetuning-Code

## /mnt/d/Projects/LLM\_finetuneing\_and\_Compression/Report/1.Completed\_Imdb\_Finetuning/GPT2-Finetuning-Code/install\_venv.sh

# Create a virtual environment named NewsAI  
python3 -m venv linuxvenv  
# Activate the virtual environment (use source command for Ubuntu)  
source linuxvenv/bin/activate  
  
# Upgrade pip within the virtual environment  
python -m pip install --upgrade pip  
  
# Install basic libraries  
pip install -r requirements.txt

## /mnt/d/Projects/LLM\_finetuneing\_and\_Compression/Report/1.Completed\_Imdb\_Finetuning/GPT2-Finetuning-Code/requirements.txt

torch   
transformers   
datasets   
matplotlib  
jupyterlab

## /mnt/d/Projects/LLM\_finetuneing\_and\_Compression/Report/1.Completed\_Imdb\_Finetuning/GPT2-Finetuning-Code/Research/1.Base.ipynb

{  
 "cells": [  
 {  
 "cell\_type": "code",  
 "execution\_count": 4,  
 "id": "b8571e12-7e04-4af4-985e-867170e1b6ee",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "import os"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 5,  
 "id": "ac545631-d8b6-4ab5-a3db-c41a5311381b",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "'/mnt/d/Projects/LLM finetuneing and Compression/Research'"  
 ]  
 },  
 "execution\_count": 5,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "%pwd"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 6,  
 "id": "b0040d75-d564-40bb-a9b9-18d42c65f357",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "os.chdir(\"../\")"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 7,  
 "id": "857c8cbd-b4e9-45a3-b0ad-fe35bc154b0c",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "'/mnt/d/Projects/LLM finetuneing and Compression'"  
 ]  
 },  
 "execution\_count": 7,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "%pwd"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 9,  
 "id": "b14ecdc0-b895-497f-a910-be2f5baee416",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "import os\n",  
 "import torch\n",  
 "import matplotlib.pyplot as plt\n",  
 "from transformers import (\n",  
 " GPT2Tokenizer, \n",  
 " GPT2LMHeadModel, \n",  
 " Trainer, \n",  
 " TrainingArguments, \n",  
 " DataCollatorForLanguageModeling,\n",  
 " TrainerCallback,\n",  
 " pipeline\n",  
 ")\n",  
 "from datasets import load\_dataset"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 10,  
 "id": "c6eac9ed-c2e6-42ca-9c8e-9a08cab9084b",  
 "metadata": {},  
 "outputs": [  
 {  
 "name": "stdout",  
 "output\_type": "stream",  
 "text": [  
 "Dataset Structure:\n",  
 "DatasetDict({\n",  
 " train: Dataset({\n",  
 " features: ['text', 'label'],\n",  
 " num\_rows: 25000\n",  
 " })\n",  
 " test: Dataset({\n",  
 " features: ['text', 'label'],\n",  
 " num\_rows: 25000\n",  
 " })\n",  
 " unsupervised: Dataset({\n",  
 " features: ['text', 'label'],\n",  
 " num\_rows: 50000\n",  
 " })\n",  
 "})\n",  
 "\n",  
 "Train dataset sample (first 5 entries):\n",  
 "{'text': 'I rented I AM CURIOUS-YELLOW from my video store because of all the controversy that surrounded it when it was first released in 1967. I also heard that at first it was seized by U.S. customs if it ever tried to enter this country, therefore being a fan of films considered \"controversial\" I really had to see this for myself.<br /><br />The plot is centered around a young Swedish drama student named Lena who wants to learn everything she can about life. In particular she wants to focus her attentions to making some sort of documentary on what the average Swede thought about certain political issues such as the Vietnam War and race issues in the United States. In between asking politicians and ordinary denizens of Stockholm about their opinions on politics, she has sex with her drama teacher, classmates, and married men.<br /><br />What kills me about I AM CURIOUS-YELLOW is that 40 years ago, this was considered pornographic. Really, the sex and nudity scenes are few and far between, even then it\\'s not shot like some cheaply made porno. While my countrymen mind find it shocking, in reality sex and nudity are a major staple in Swedish cinema. Even Ingmar Bergman, arguably their answer to good old boy John Ford, had sex scenes in his films.<br /><br />I do commend the filmmakers for the fact that any sex shown in the film is shown for artistic purposes rather than just to shock people and make money to be shown in pornographic theaters in America. I AM CURIOUS-YELLOW is a good film for anyone wanting to study the meat and potatoes (no pun intended) of Swedish cinema. But really, this film doesn\\'t have much of a plot.', 'label': 0}\n",  
 "{'text': '\"I Am Curious: Yellow\" is a risible and pretentious steaming pile. It doesn\\'t matter what one\\'s political views are because this film can hardly be taken seriously on any level. As for the claim that frontal male nudity is an automatic NC-17, that isn\\'t true. I\\'ve seen R-rated films with male nudity. Granted, they only offer some fleeting views, but where are the R-rated films with gaping vulvas and flapping labia? Nowhere, because they don\\'t exist. The same goes for those crappy cable shows: schlongs swinging in the breeze but not a clitoris in sight. And those pretentious indie movies like The Brown Bunny, in which we\\'re treated to the site of Vincent Gallo\\'s throbbing johnson, but not a trace of pink visible on Chloe Sevigny. Before crying (or implying) \"double-standard\" in matters of nudity, the mentally obtuse should take into account one unavoidably obvious anatomical difference between men and women: there are no genitals on display when actresses appears nude, and the same cannot be said for a man. In fact, you generally won\\'t see female genitals in an American film in anything short of porn or explicit erotica. This alleged double-standard is less a double standard than an admittedly depressing ability to come to terms culturally with the insides of women\\'s bodies.', 'label': 0}\n",  
 "{'text': \"If only to avoid making this type of film in the future. This film is interesting as an experiment but tells no cogent story.<br /><br />One might feel virtuous for sitting thru it because it touches on so many IMPORTANT issues but it does so without any discernable motive. The viewer comes away with no new perspectives (unless one comes up with one while one's mind wanders, as it will invariably do during this pointless film).<br /><br />One might better spend one's time staring out a window at a tree growing.<br /><br />\", 'label': 0}\n",  
 "{'text': \"This film was probably inspired by Godard's Masculin, féminin and I urge you to see that film instead.<br /><br />The film has two strong elements and those are, (1) the realistic acting (2) the impressive, undeservedly good, photo. Apart from that, what strikes me most is the endless stream of silliness. Lena Nyman has to be most annoying actress in the world. She acts so stupid and with all the nudity in this film,...it's unattractive. Comparing to Godard's film, intellectuality has been replaced with stupidity. Without going too far on this subject, I would say that follows from the difference in ideals between the French and the Swedish society.<br /><br />A movie of its time, and place. 2/10.\", 'label': 0}\n",  
 "{'text': 'Oh, brother...after hearing about this ridiculous film for umpteen years all I can think of is that old Peggy Lee song..<br /><br />\"Is that all there is??\" ...I was just an early teen when this smoked fish hit the U.S. I was too young to get in the theater (although I did manage to sneak into \"Goodbye Columbus\"). Then a screening at a local film museum beckoned - Finally I could see this film, except now I was as old as my parents were when they schlepped to see it!!<br /><br />The ONLY reason this film was not condemned to the anonymous sands of time was because of the obscenity case sparked by its U.S. release. MILLIONS of people flocked to this stinker, thinking they were going to see a sex film...Instead, they got lots of closeups of gnarly, repulsive Swedes, on-street interviews in bland shopping malls, asinie political pretension...and feeble who-cares simulated sex scenes with saggy, pale actors.<br /><br />Cultural icon, holy grail, historic artifact..whatever this thing was, shred it, burn it, then stuff the ashes in a lead box!<br /><br />Elite esthetes still scrape to find value in its boring pseudo revolutionary political spewings..But if it weren\\'t for the censorship scandal, it would have been ignored, then forgotten.<br /><br />Instead, the \"I Am Blank, Blank\" rhythymed title was repeated endlessly for years as a titilation for porno films (I am Curious, Lavender - for gay films, I Am Curious, Black - for blaxploitation films, etc..) and every ten years or so the thing rises from the dead, to be viewed by a new generation of suckers who want to see that \"naughty sex film\" that \"revolutionized the film industry\"...<br /><br />Yeesh, avoid like the plague..Or if you MUST see it - rent the video and fast forward to the \"dirty\" parts, just to get it over with.<br /><br />', 'label': 0}\n",  
 "\n",  
 "Test dataset sample (first 5 entries):\n",  
 "{'text': 'I love sci-fi and am willing to put up with a lot. Sci-fi movies/TV are usually underfunded, under-appreciated and misunderstood. I tried to like this, I really did, but it is to good TV sci-fi as Babylon 5 is to Star Trek (the original). Silly prosthetics, cheap cardboard sets, stilted dialogues, CG that doesn\\'t match the background, and painfully one-dimensional characters cannot be overcome with a \\'sci-fi\\' setting. (I\\'m sure there are those of you out there who think Babylon 5 is good sci-fi TV. It\\'s not. It\\'s clichéd and uninspiring.) While US viewers might like emotion and character development, sci-fi is a genre that does not take itself seriously (cf. Star Trek). It may treat important issues, yet not as a serious philosophy. It\\'s really difficult to care about the characters here as they are not simply foolish, just missing a spark of life. Their actions and reactions are wooden and predictable, often painful to watch. The makers of Earth KNOW it\\'s rubbish as they have to always say \"Gene Roddenberry\\'s Earth...\" otherwise people would not continue watching. Roddenberry\\'s ashes must be turning in their orbit as this dull, cheap, poorly edited (watching it without advert breaks really brings this home) trudging Trabant of a show lumbers into space. Spoiler. So, kill off a main character. And then bring him back as another actor. Jeeez! Dallas all over again.', 'label': 0}\n",  
 "{'text': \"Worth the entertainment value of a rental, especially if you like action movies. This one features the usual car chases, fights with the great Van Damme kick style, shooting battles with the 40 shell load shotgun, and even terrorist style bombs. All of this is entertaining and competently handled but there is nothing that really blows you away if you've seen your share before.<br /><br />The plot is made interesting by the inclusion of a rabbit, which is clever but hardly profound. Many of the characters are heavily stereotyped -- the angry veterans, the terrified illegal aliens, the crooked cops, the indifferent feds, the bitchy tough lady station head, the crooked politician, the fat federale who looks like he was typecast as the Mexican in a Hollywood movie from the 1940s. All passably acted but again nothing special.<br /><br />I thought the main villains were pretty well done and fairly well acted. By the end of the movie you certainly knew who the good guys were and weren't. There was an emotional lift as the really bad ones got their just deserts. Very simplistic, but then you weren't expecting Hamlet, right? The only thing I found really annoying was the constant cuts to VDs daughter during the last fight scene.<br /><br />Not bad. Not good. Passable 4.\", 'label': 0}\n",  
 "{'text': \"its a totally average film with a few semi-alright action sequences that make the plot seem a little better and remind the viewer of the classic van dam films. parts of the plot don't make sense and seem to be added in to use up time. the end plot is that of a very basic type that doesn't leave the viewer guessing and any twists are obvious from the beginning. the end scene with the flask backs don't make sense as they are added in and seem to have little relevance to the history of van dam's character. not really worth watching again, bit disappointed in the end production, even though it is apparent it was shot on a low budget certain shots and sections in the film are of poor directed quality\", 'label': 0}\n",  
 "{'text': \"STAR RATING: \*\*\*\*\* Saturday Night \*\*\*\* Friday Night \*\*\* Friday Morning \*\* Sunday Night \* Monday Morning <br /><br />Former New Orleans homicide cop Jack Robideaux (Jean Claude Van Damme) is re-assigned to Columbus, a small but violent town in Mexico to help the police there with their efforts to stop a major heroin smuggling operation into their town. The culprits turn out to be ex-military, lead by former commander Benjamin Meyers (Stephen Lord, otherwise known as Jase from East Enders) who is using a special method he learned in Afghanistan to fight off his opponents. But Jack has a more personal reason for taking him down, that draws the two men into an explosive final showdown where only one will walk away alive.<br /><br />After Until Death, Van Damme appeared to be on a high, showing he could make the best straight to video films in the action market. While that was a far more drama oriented film, with The Shepherd he has returned to the high-kicking, no brainer action that first made him famous and has sadly produced his worst film since Derailed. It's nowhere near as bad as that film, but what I said still stands.<br /><br />A dull, predictable film, with very little in the way of any exciting action. What little there is mainly consists of some limp fight scenes, trying to look cool and trendy with some cheap slo-mo/sped up effects added to them that sadly instead make them look more desperate. Being a Mexican set film, director Isaac Florentine has tried to give the film a Robert Rodriguez/Desperado sort of feel, but this only adds to the desperation.<br /><br />VD gives a particularly uninspired performance and given he's never been a Robert De Niro sort of actor, that can't be good. As the villain, Lord shouldn't expect to leave the beeb anytime soon. He gets little dialogue at the beginning as he struggles to muster an American accent but gets mysteriously better towards the end. All the supporting cast are equally bland, and do nothing to raise the films spirits at all.<br /><br />This is one shepherd that's strayed right from the flock. \*\", 'label': 0}\n",  
 "{'text': \"First off let me say, If you haven't enjoyed a Van Damme movie since bloodsport, you probably will not like this movie. Most of these movies may not have the best plots or best actors but I enjoy these kinds of movies for what they are. This movie is much better than any of the movies the other action guys (Segal and Dolph) have thought about putting out the past few years. Van Damme is good in the movie, the movie is only worth watching to Van Damme fans. It is not as good as Wake of Death (which i highly recommend to anyone of likes Van Damme) or In hell but, in my opinion it's worth watching. It has the same type of feel to it as Nowhere to Run. Good fun stuff!\", 'label': 0}\n"  
 ]  
 }  
 ],  
 "source": [  
 "#####################################\n",  
 "# 1. Imports and Dataset Loading\n",  
 "#####################################\n",  
 "\n",  
 "# Load the IMDb dataset from the Hugging Face hub\n",  
 "dataset = load\_dataset(\"imdb\")\n",  
 "\n",  
 "# Display the dataset structure\n",  
 "print(\"Dataset Structure:\")\n",  
 "print(dataset)\n",  
 "\n",  
 "# Print the first 5 lines of the train and test datasets\n",  
 "print(\"\\nTrain dataset sample (first 5 entries):\")\n",  
 "for i in range(5):\n",  
 " print(dataset[\"train\"][i])\n",  
 "\n",  
 "print(\"\\nTest dataset sample (first 5 entries):\")\n",  
 "for i in range(5):\n",  
 " print(dataset[\"test\"][i])\n"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 11,  
 "id": "f813a551-7509-4d89-8140-6186147a7ab7",  
 "metadata": {},  
 "outputs": [  
 {  
 "name": "stderr",  
 "output\_type": "stream",  
 "text": [  
 "Device set to use cuda:0\n",  
 "Truncation was not explicitly activated but `max\_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting to 'longest\_first' truncation strategy. If you encode pairs of sequences (GLUE-style) with the tokenizer you can select this strategy more precisely by providing a specific strategy to `truncation`.\n"  
 ]  
 },  
 {  
 "name": "stdout",  
 "output\_type": "stream",  
 "text": [  
 "\n",  
 "Base model text generation:\n",  
 "Once upon a time my voice had risen, I trembled, as if on a very high stage. When I heard the voice of my own ears, I stopped and rose. I am now fully aware of my identity; I was born a man\n"  
 ]  
 }  
 ],  
 "source": [  
 "#####################################\n",  
 "# 2. Save the Base Model\n",  
 "#####################################\n",  
 "\n",  
 "# Initialize GPT-2 tokenizer and set padding token (GPT-2 doesn't have one by default)\n",  
 "tokenizer = GPT2Tokenizer.from\_pretrained(\"gpt2\")\n",  
 "tokenizer.pad\_token = tokenizer.eos\_token\n",  
 "\n",  
 "# Load the GPT-2 model with a language modeling head\n",  
 "model = GPT2LMHeadModel.from\_pretrained(\"gpt2\")\n",  
 "\n",  
 "# Create directory for saving if it doesn't exist\n",  
 "os.makedirs(\"./model/gpt2-base\", exist\_ok=True)\n",  
 "\n",  
 "# Save the base model and tokenizer\n",  
 "model.save\_pretrained(\"./model/gpt2-base\")\n",  
 "tokenizer.save\_pretrained(\"./model/gpt2-base\")\n",  
 "\n",  
 "#####################################\n",  
 "# 3. Generate Text with Base Model\n",  
 "#####################################\n",  
 "\n",  
 "# Initialize a text generation pipeline using the base model\n",  
 "generator = pipeline('text-generation', model=model, tokenizer=tokenizer)\n",  
 "\n",  
 "# Generate sample text\n",  
 "print(\"\\nBase model text generation:\")\n",  
 "generated\_text = generator(\"Once upon a time\", max\_length=50, num\_return\_sequences=1)[0][\"generated\_text\"]\n",  
 "print(generated\_text)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 12,  
 "id": "a85a8c34-4807-44dd-b85b-b2eab350cac1",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "application/vnd.jupyter.widget-view+json": {  
 "model\_id": "79af65064e1a4d43b62b6ad686fae1e9",  
 "version\_major": 2,  
 "version\_minor": 0  
 },  
 "text/plain": [  
 "Map: 0%| | 0/25000 [00:00<?, ? examples/s]"  
 ]  
 },  
 "metadata": {},  
 "output\_type": "display\_data"  
 },  
 {  
 "data": {  
 "application/vnd.jupyter.widget-view+json": {  
 "model\_id": "3ba1cb94f2b34bd2a0eeab6b91bd51e8",  
 "version\_major": 2,  
 "version\_minor": 0  
 },  
 "text/plain": [  
 "Map: 0%| | 0/25000 [00:00<?, ? examples/s]"  
 ]  
 },  
 "metadata": {},  
 "output\_type": "display\_data"  
 },  
 {  
 "data": {  
 "application/vnd.jupyter.widget-view+json": {  
 "model\_id": "05ab93056f5545dcae9efb97b5b320b9",  
 "version\_major": 2,  
 "version\_minor": 0  
 },  
 "text/plain": [  
 "Map: 0%| | 0/50000 [00:00<?, ? examples/s]"  
 ]  
 },  
 "metadata": {},  
 "output\_type": "display\_data"  
 }  
 ],  
 "source": [  
 "#####################################\n",  
 "# 4. Fine-Tuning Preparation\n",  
 "#####################################\n",  
 "\n",  
 "# Define a function to tokenize text data\n",  
 "def tokenize\_function(examples):\n",  
 " # Tokenize each review with a maximum sequence length. Adjust max\_length as needed.\n",  
 " return tokenizer(examples[\"text\"], truncation=True, max\_length=512)\n",  
 "\n",  
 "# Tokenize the dataset; remove columns that are not needed (here we remove \"text\" and \"label\")\n",  
 "tokenized\_datasets = dataset.map(tokenize\_function, batched=True, remove\_columns=[\"text\", \"label\"])\n",  
 "\n",  
 "# Data collator that batches sequences and pads them dynamically.\n",  
 "# Since GPT-2 is a causal language model, we set mlm=False.\n",  
 "data\_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 13,  
 "id": "a7f0814e-6309-4fc0-8618-03c6893adf53",  
 "metadata": {},  
 "outputs": [  
 {  
 "name": "stderr",  
 "output\_type": "stream",  
 "text": [  
 "/home/ayushman/.local/lib/python3.10/site-packages/transformers/training\_args.py:1594: FutureWarning: `evaluation\_strategy` is deprecated and will be removed in version 4.46 of 🤗 Transformers. Use `eval\_strategy` instead\n",  
 " warnings.warn(\n",  
 "`loss\_type=None` was set in the config but it is unrecognised.Using the default loss: `ForCausalLMLoss`.\n"  
 ]  
 },  
 {  
 "data": {  
 "text/html": [  
 "\n",  
 " <div>\n",  
 " \n",  
 " <progress value='37500' max='37500' style='width:300px; height:20px; vertical-align: middle;'></progress>\n",  
 " [37500/37500 6:04:58, Epoch 3/3]\n",  
 " </div>\n",  
 " <table border=\"1\" class=\"dataframe\">\n",  
 " <thead>\n",  
 " <tr style=\"text-align: left;\">\n",  
 " <th>Step</th>\n",  
 " <th>Training Loss</th>\n",  
 " <th>Validation Loss</th>\n",  
 " </tr>\n",  
 " </thead>\n",  
 " <tbody>\n",  
 " <tr>\n",  
 " <td>500</td>\n",  
 " <td>3.672200</td>\n",  
 " <td>3.605307</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>1000</td>\n",  
 " <td>3.717400</td>\n",  
 " <td>3.588417</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>1500</td>\n",  
 " <td>3.669500</td>\n",  
 " <td>3.570712</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>2000</td>\n",  
 " <td>3.628100</td>\n",  
 " <td>3.563755</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>2500</td>\n",  
 " <td>3.591800</td>\n",  
 " <td>3.557327</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>3000</td>\n",  
 " <td>3.610500</td>\n",  
 " <td>3.550287</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>3500</td>\n",  
 " <td>3.609900</td>\n",  
 " <td>3.543281</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>4000</td>\n",  
 " <td>3.653800</td>\n",  
 " <td>3.535744</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>4500</td>\n",  
 " <td>3.651000</td>\n",  
 " <td>3.534860</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>5000</td>\n",  
 " <td>3.543000</td>\n",  
 " <td>3.530080</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>5500</td>\n",  
 " <td>3.617600</td>\n",  
 " <td>3.525420</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>6000</td>\n",  
 " <td>3.612000</td>\n",  
 " <td>3.518645</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>6500</td>\n",  
 " <td>3.532900</td>\n",  
 " <td>3.518119</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>7000</td>\n",  
 " <td>3.631900</td>\n",  
 " <td>3.512174</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>7500</td>\n",  
 " <td>3.565600</td>\n",  
 " <td>3.508458</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>8000</td>\n",  
 " <td>3.579700</td>\n",  
 " <td>3.506429</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>8500</td>\n",  
 " <td>3.547800</td>\n",  
 " <td>3.503904</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>9000</td>\n",  
 " <td>3.590200</td>\n",  
 " <td>3.499748</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>9500</td>\n",  
 " <td>3.573500</td>\n",  
 " <td>3.497056</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>10000</td>\n",  
 " <td>3.541800</td>\n",  
 " <td>3.495464</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>10500</td>\n",  
 " <td>3.525500</td>\n",  
 " <td>3.494889</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>11000</td>\n",  
 " <td>3.497900</td>\n",  
 " <td>3.489882</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>11500</td>\n",  
 " <td>3.530400</td>\n",  
 " <td>3.486683</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>12000</td>\n",  
 " <td>3.524100</td>\n",  
 " <td>3.485568</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>12500</td>\n",  
 " <td>3.522400</td>\n",  
 " <td>3.482388</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>13000</td>\n",  
 " <td>3.416000</td>\n",  
 " <td>3.484618</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>13500</td>\n",  
 " <td>3.368500</td>\n",  
 " <td>3.486907</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>14000</td>\n",  
 " <td>3.394100</td>\n",  
 " <td>3.485422</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>14500</td>\n",  
 " <td>3.405000</td>\n",  
 " <td>3.486501</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>15000</td>\n",  
 " <td>3.360000</td>\n",  
 " <td>3.483894</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>15500</td>\n",  
 " <td>3.379200</td>\n",  
 " <td>3.480854</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>16000</td>\n",  
 " <td>3.410900</td>\n",  
 " <td>3.480395</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>16500</td>\n",  
 " <td>3.377900</td>\n",  
 " <td>3.481333</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>17000</td>\n",  
 " <td>3.375400</td>\n",  
 " <td>3.478402</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>17500</td>\n",  
 " <td>3.401300</td>\n",  
 " <td>3.477528</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>18000</td>\n",  
 " <td>3.418300</td>\n",  
 " <td>3.474829</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>18500</td>\n",  
 " <td>3.383300</td>\n",  
 " <td>3.473990</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>19000</td>\n",  
 " <td>3.426100</td>\n",  
 " <td>3.472247</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>19500</td>\n",  
 " <td>3.425400</td>\n",  
 " <td>3.470489</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>20000</td>\n",  
 " <td>3.415300</td>\n",  
 " <td>3.469525</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>20500</td>\n",  
 " <td>3.363200</td>\n",  
 " <td>3.467826</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>21000</td>\n",  
 " <td>3.334500</td>\n",  
 " <td>3.469357</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>21500</td>\n",  
 " <td>3.414800</td>\n",  
 " <td>3.467202</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>22000</td>\n",  
 " <td>3.380700</td>\n",  
 " <td>3.467320</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>22500</td>\n",  
 " <td>3.407200</td>\n",  
 " <td>3.465333</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>23000</td>\n",  
 " <td>3.400400</td>\n",  
 " <td>3.463091</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>23500</td>\n",  
 " <td>3.355800</td>\n",  
 " <td>3.462481</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>24000</td>\n",  
 " <td>3.389800</td>\n",  
 " <td>3.461218</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>24500</td>\n",  
 " <td>3.353500</td>\n",  
 " <td>3.458194</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>25000</td>\n",  
 " <td>3.352700</td>\n",  
 " <td>3.458542</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>25500</td>\n",  
 " <td>3.256700</td>\n",  
 " <td>3.466905</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>26000</td>\n",  
 " <td>3.261300</td>\n",  
 " <td>3.466107</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>26500</td>\n",  
 " <td>3.216700</td>\n",  
 " <td>3.467902</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>27000</td>\n",  
 " <td>3.256300</td>\n",  
 " <td>3.466283</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>27500</td>\n",  
 " <td>3.276500</td>\n",  
 " <td>3.465291</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>28000</td>\n",  
 " <td>3.282900</td>\n",  
 " <td>3.463394</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>28500</td>\n",  
 " <td>3.273500</td>\n",  
 " <td>3.463842</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>29000</td>\n",  
 " <td>3.320000</td>\n",  
 " <td>3.461794</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>29500</td>\n",  
 " <td>3.275300</td>\n",  
 " <td>3.462550</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>30000</td>\n",  
 " <td>3.296300</td>\n",  
 " <td>3.460799</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>30500</td>\n",  
 " <td>3.306300</td>\n",  
 " <td>3.460644</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>31000</td>\n",  
 " <td>3.271500</td>\n",  
 " <td>3.459811</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>31500</td>\n",  
 " <td>3.215700</td>\n",  
 " <td>3.463105</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>32000</td>\n",  
 " <td>3.230500</td>\n",  
 " <td>3.459745</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>32500</td>\n",  
 " <td>3.281400</td>\n",  
 " <td>3.457821</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>33000</td>\n",  
 " <td>3.255900</td>\n",  
 " <td>3.458639</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>33500</td>\n",  
 " <td>3.261600</td>\n",  
 " <td>3.456487</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>34000</td>\n",  
 " <td>3.271800</td>\n",  
 " <td>3.456322</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>34500</td>\n",  
 " <td>3.251100</td>\n",  
 " <td>3.455962</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>35000</td>\n",  
 " <td>3.269100</td>\n",  
 " <td>3.454641</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>35500</td>\n",  
 " <td>3.272100</td>\n",  
 " <td>3.455085</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>36000</td>\n",  
 " <td>3.255300</td>\n",  
 " <td>3.454105</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>36500</td>\n",  
 " <td>3.254800</td>\n",  
 " <td>3.454243</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>37000</td>\n",  
 " <td>3.269300</td>\n",  
 " <td>3.454247</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <td>37500</td>\n",  
 " <td>3.303800</td>\n",  
 " <td>3.453540</td>\n",  
 " </tr>\n",  
 " </tbody>\n",  
 "</table><p>"  
 ],  
 "text/plain": [  
 "<IPython.core.display.HTML object>"  
 ]  
 },  
 "metadata": {},  
 "output\_type": "display\_data"  
 },  
 {  
 "data": {  
 "image/png": "",  
 "text/plain": [  
 "<Figure size 640x480 with 1 Axes>"  
 ]  
 },  
 "metadata": {},  
 "output\_type": "display\_data"  
 }  
 ],  
 "source": [  
 "#####################################\n",  
 "# 5. Fine-Tuning with Training Graph Metrics\n",  
 "#####################################\n",  
 "\n",  
 "from transformers import TrainerCallback # Ensure TrainerCallback is imported\n",  
 "\n",  
 "# Define training arguments. Adjust parameters (epochs, batch size, etc.) as needed.\n",  
 "training\_args = TrainingArguments(\n",  
 " output\_dir=\"./model/gpt2-imdb-finetuned\",\n",  
 " overwrite\_output\_dir=True,\n",  
 " num\_train\_epochs=3,\n",  
 " per\_device\_train\_batch\_size=2,\n",  
 " per\_device\_eval\_batch\_size=2,\n",  
 " save\_steps=500,\n",  
 " # evaluation\_strategy=\"steps\",\n",  
 " eval\_strategy=\"steps\", # Updated argument name\n",  
 " eval\_steps=500,\n",  
 " save\_total\_limit=2,\n",  
 " logging\_steps=100,\n",  
 " prediction\_loss\_only=True,\n",  
 " fp16=torch.cuda.is\_available(), # Use FP16 if available for faster training\n",  
 " report\_to=\"none\", # disable reporting integrations (e.g., wandb)\n",  
 ")\n",  
 "\n",  
 "# Custom callback to log training steps and loss for plotting\n",  
 "class LogCallback(TrainerCallback):\n",  
 " def \_\_init\_\_(self):\n",  
 " self.logs = []\n",  
 " def on\_log(self, args, state, control, logs=None, \*\*kwargs):\n",  
 " if logs is not None and \"loss\" in logs:\n",  
 " self.logs.append((state.global\_step, logs[\"loss\"]))\n",  
 "\n",  
 "log\_callback = LogCallback()\n",  
 "\n",  
 "# (Optional) Re-load the base model for fine-tuning if not already loaded/modified\n",  
 "model = GPT2LMHeadModel.from\_pretrained(\"gpt2\")\n",  
 "\n",  
 "# Initialize the Trainer with the model, training arguments, datasets, data collator, and our logging callback\n",  
 "trainer = Trainer(\n",  
 " model=model,\n",  
 " args=training\_args,\n",  
 " data\_collator=data\_collator,\n",  
 " train\_dataset=tokenized\_datasets[\"train\"],\n",  
 " eval\_dataset=tokenized\_datasets[\"test\"],\n",  
 " callbacks=[log\_callback],\n",  
 ")\n",  
 "\n",  
 "# Start fine-tuning\n",  
 "trainer.train()\n",  
 "\n",  
 "# Save the final model and tokenizer\n",  
 "model.save\_pretrained(\"./model/gpt2-imdb-finetuned\")\n",  
 "tokenizer.save\_pretrained(\"./model/gpt2-imdb-finetuned\")\n",  
 "\n",  
 "# Plot the training loss graph using the logged metrics\n",  
 "if log\_callback.logs:\n",  
 " steps, losses = zip(\*log\_callback.logs)\n",  
 " import matplotlib.pyplot as plt\n",  
 " plt.figure()\n",  
 " plt.plot(steps, losses, marker='o')\n",  
 " plt.xlabel(\"Training Steps\")\n",  
 " plt.ylabel(\"Loss\")\n",  
 " plt.title(\"Training Loss Over Time\")\n",  
 " plt.grid(True)\n",  
 " plt.show()\n",  
 "else:\n",  
 " print(\"No logs were recorded.\")"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": null,  
 "id": "2dbe3d90-dab6-4011-b6bd-f13065d862f3",  
 "metadata": {},  
 "outputs": [],  
 "source": []  
 }  
 ],  
 "metadata": {  
 "kernelspec": {  
 "display\_name": "Python 3 (ipykernel)",  
 "language": "python",  
 "name": "python3"  
 },  
 "language\_info": {  
 "codemirror\_mode": {  
 "name": "ipython",  
 "version": 3  
 },  
 "file\_extension": ".py",  
 "mimetype": "text/x-python",  
 "name": "python",  
 "nbconvert\_exporter": "python",  
 "pygments\_lexer": "ipython3",  
 "version": "3.10.12"  
 }  
 },  
 "nbformat": 4,  
 "nbformat\_minor": 5  
}

## /mnt/d/Projects/LLM\_finetuneing\_and\_Compression/Report/1.Completed\_Imdb\_Finetuning/GPT2-Finetuning-Code/Research/1.Calling base Model.ipynb

{  
 "cells": [  
 {  
 "cell\_type": "code",  
 "execution\_count": 1,  
 "id": "0727b9ab-4b1f-4ed0-9fdd-9c9b677ca030",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "import os"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 2,  
 "id": "568e9732-482e-4617-b6b3-87a3b9956672",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "'/mnt/d/Projects/LLM finetuneing and Compression/Research'"  
 ]  
 },  
 "execution\_count": 2,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "%pwd"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 3,  
 "id": "e8d0522a-3ae8-4856-9912-c7e819964412",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "os.chdir(\"../\")"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 4,  
 "id": "7f10fe2b-6337-4af1-b8c3-b15ce7276878",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "'/mnt/d/Projects/LLM finetuneing and Compression'"  
 ]  
 },  
 "execution\_count": 4,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "%pwd"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 10,  
 "id": "17cfd411-d27f-4643-ac7b-723dd09de1fa",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "from transformers import (\n",  
 " GPT2Tokenizer, \n",  
 " GPT2LMHeadModel, \n",  
 " # Trainer, \n",  
 " # TrainingArguments, \n",  
 " # DataCollatorForLanguageModeling,\n",  
 " # TrainerCallback,\n",  
 " pipeline\n",  
 ")"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 11,  
 "id": "49cf4f7d-0417-4cd3-8016-27cd07ba1273",  
 "metadata": {},  
 "outputs": [  
 {  
 "name": "stderr",  
 "output\_type": "stream",  
 "text": [  
 "Device set to use cuda:0\n",  
 "Truncation was not explicitly activated but `max\_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting to 'longest\_first' truncation strategy. If you encode pairs of sequences (GLUE-style) with the tokenizer you can select this strategy more precisely by providing a specific strategy to `truncation`.\n"  
 ]  
 },  
 {  
 "name": "stdout",  
 "output\_type": "stream",  
 "text": [  
 "\n",  
 "Base model text generation:\n",  
 "Once upon a time, two guys had a mutual fondness in one place for one another after being in love and having a bad life. Their first time together was at the age of 12 and just before they got divorced at a party. But when\n"  
 ]  
 }  
 ],  
 "source": [  
 "#####################################\n",  
 "# 2. Calling finetuned model \n",  
 "#####################################\n",  
 "\n",  
 "# Initialize GPT-2 tokenizer and set padding token (GPT-2 doesn't have one by default)\n",  
 "tokenizer = GPT2Tokenizer.from\_pretrained(\"model/gpt2-imdb-finetuned\")\n",  
 "tokenizer.pad\_token = tokenizer.eos\_token\n",  
 "\n",  
 "# Load the GPT-2 model with a language modeling head\n",  
 "model = GPT2LMHeadModel.from\_pretrained(\"model/gpt2-imdb-finetuned\")\n",  
 "\n",  
 "#####################################\n",  
 "# 3. Generate Text with Base Model\n",  
 "#####################################\n",  
 "\n",  
 "# Initialize a text generation pipeline using the base model\n",  
 "generator = pipeline('text-generation', model=model, tokenizer=tokenizer)\n",  
 "\n",  
 "# Generate sample text\n",  
 "print(\"\\nBase model text generation:\")\n",  
 "generated\_text = generator(\"Once upon a time\", max\_length=50, num\_return\_sequences=1)[0][\"generated\_text\"]\n",  
 "print(generated\_text)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 12,  
 "id": "643575dc-1229-46d5-a2cb-d309c7c6a1e8",  
 "metadata": {},  
 "outputs": [  
 {  
 "name": "stderr",  
 "output\_type": "stream",  
 "text": [  
 "Device set to use cuda:0\n",  
 "Truncation was not explicitly activated but `max\_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting to 'longest\_first' truncation strategy. If you encode pairs of sequences (GLUE-style) with the tokenizer you can select this strategy more precisely by providing a specific strategy to `truncation`.\n"  
 ]  
 },  
 {  
 "name": "stdout",  
 "output\_type": "stream",  
 "text": [  
 "\n",  
 "Base model text generation:\n",  
 "Once upon a time or a few years these people would have been going up there with their own business. They would have been taking a small slice of American jobs, getting their own homes, and living at home. They would feel comfortable spending money on\n"  
 ]  
 }  
 ],  
 "source": [  
 "#####################################\n",  
 "# 2. Calling base model\n",  
 "#####################################\n",  
 "\n",  
 "# Initialize GPT-2 tokenizer and set padding token (GPT-2 doesn't have one by default)\n",  
 "tokenizer = GPT2Tokenizer.from\_pretrained(\"model/gpt2-base\")\n",  
 "tokenizer.pad\_token = tokenizer.eos\_token\n",  
 "\n",  
 "# Load the GPT-2 model with a language modeling head\n",  
 "model = GPT2LMHeadModel.from\_pretrained(\"model/gpt2-base\")\n",  
 "\n",  
 "#####################################\n",  
 "# 3. Generate Text with Base Model\n",  
 "#####################################\n",  
 "\n",  
 "# Initialize a text generation pipeline using the base model\n",  
 "generator = pipeline('text-generation', model=model, tokenizer=tokenizer)\n",  
 "\n",  
 "# Generate sample text\n",  
 "print(\"\\nBase model text generation:\")\n",  
 "generated\_text = generator(\"Once upon a time\", max\_length=50, num\_return\_sequences=1)[0][\"generated\_text\"]\n",  
 "print(generated\_text)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": null,  
 "id": "22b8cd8d-c04d-47c0-a456-4bdd26038a3a",  
 "metadata": {},  
 "outputs": [],  
 "source": []  
 }  
 ],  
 "metadata": {  
 "kernelspec": {  
 "display\_name": "Python 3 (ipykernel)",  
 "language": "python",  
 "name": "python3"  
 },  
 "language\_info": {  
 "codemirror\_mode": {  
 "name": "ipython",  
 "version": 3  
 },  
 "file\_extension": ".py",  
 "mimetype": "text/x-python",  
 "name": "python",  
 "nbconvert\_exporter": "python",  
 "pygments\_lexer": "ipython3",  
 "version": "3.10.12"  
 }  
 },  
 "nbformat": 4,  
 "nbformat\_minor": 5  
}