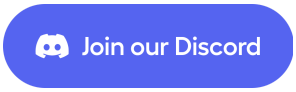
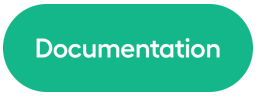
To run this, press "*Runtime*" and press "*Run all*" on a **free** Tesla T4 Google Colab instance!

(https://unsloth.ai/)



(https://discord.gg/unsloth)



(https://docs.unsloth.ai/) Join Discord if you need help + ⭐ *Star us on Github (https://github.com/unslothai/unsloth)* ⭐

To install Unsloth your local device, follow our guide (https://docs.unsloth.ai/get-started/install-and-update). This notebook is licensed LGPL-3.0 (https://github.com/unslothai/notebooks?tab=LGPL-3.0-1-ov-file#readme).

You will learn how to do data prep, how to train, how to run the model, & how to save it

News

Unsloth's Docker image (https://hub.docker.com/r/unsloth/unsloth) is here! Start training with no setup & environment issues. Read our Guide (https://docs.unsloth.ai/new/how-to-train-llms-with-unsloth-and-docker).

gpt-oss RL (https://docs.unsloth.ai/new/gpt-oss-reinforcement-learning) is now supported with the fastest inference & lowest VRAM. Try our new notebook (https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/gpt-oss-(20B)-GRPO.ipynb) which creates kernels!

Introducing Vision (https://docs.unsloth.ai/new/vision-reinforcement-learning-vlm-rl) and Standby (https://docs.unsloth.ai/basics/memory-efficient-rl) for RL! Train Qwen, Gemma etc. VLMs with GSPO - even faster with less VRAM.

Unsloth now supports Text-to-Speech (TTS) models. Read our guide here (https://docs.unsloth.ai/basics/text-to-speech-tts-fine-tuning).

Visit our docs for all our model uploads (https://docs.unsloth.ai/get-started/all-our-models) and notebooks (https://docs.unsloth.ai/get-started/unsloth-notebooks).

Installation

In [1]:
```
%%capture
import os, importlib.util
!pip install --upgrade -qqq uv
if importlib.util.find_spec("torch") is None or "COLAB_" in "".join(os.environ.keys()):
    try: import numpy, PIL; get_numpy = f"numpy=={numpy.__version__}"; get_pil = f"pillow=={PIL.__version__}"
    except: get_numpy = "numpy"; get_pil = "pillow"
    !uv pip install -qqq \
        "torch>=2.8.0" "triton>=3.4.0" {get_numpy} {get_pil} torchvision bitsandbytes "transformers==4.56.2" \
        "unsloth_zoo[base] @ git+https://github.com/unslothai/unsloth-zoo" \
        "unsloth[base] @ git+https://github.com/unslothai/unsloth" \
        git+https://github.com/triton-lang/triton.git@05b2c186c1b6c9a08375389d5efe9cb4c401c075#subdirectory=python/triton_kernels
elif importlib.util.find_spec("unsloth") is None:
    !uv pip install -qqq unsloth
!uv pip install --upgrade --no-deps transformers==4.56.2 tokenizers trl==0.22.2 unsloth unsloth_zoo
```

Unsloth

We're about to demonstrate the power of the new OpenAI GPT-OSS 20B model through a finetuning example. To use our `MXFP4` inference example, use this [notebook](https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/GPT_OSS_MXFP4_(20B)-Inference.ipynb) instead.

In [2]:
```
from unsloth import FastLanguageModel
import torch
max_seq_length = 1024
dtype = None

# 4bit pre quantized models we support for 4x faster downloading + no OOMs.
fourbit_models = [
    "unsloth/gpt-oss-20b-unsloth-bnb-4bit", # 20B model using bitsandbytes 4bi
t quantization
    "unsloth/gpt-oss-120b-unsloth-bnb-4bit",
    "unsloth/gpt-oss-20b", # 20B model using MXFP4 format
    "unsloth/gpt-oss-120b",
] # More models at https://huggingface.co/unsloth

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/gpt-oss-20b",
    dtype = dtype, # None for auto detection
    max_seq_length = max_seq_length, # Choose any for long context!
    load_in_4bit = True,  # 4 bit quantization to reduce memory
    full_finetuning = False, # [NEW!] We have full finetuning now!
    # token = "hf_...", # use one if using gated models
)
```

```
🦥 Unsloth: Will patch your computer to enable 2x faster free finetuning.
🦥 Unsloth Zoo will now patch everything to make training faster!
==((====))==  Unsloth 2025.11.3: Fast Gpt_Oss patching. Transformers: 4.56.2.
   \\   /|    Tesla T4. Num GPUs = 1. Max memory: 14.741 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.8.0+cu126. CUDA: 7.5. CUDA Toolkit: 12.6. Triton: 3.4.
0
\        /    Bfloat16 = FALSE. FA [Xformers = None. FA2 = False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red
colored!
Unsloth: Using float16 precision for gpt_oss won't work! Using float32.
```

We now add LoRA adapters for parameter efficient finetuning - this allows us to only efficiently train 1% of all parameters.

In [3]:
```
model = FastLanguageModel.get_peft_model(
    model,
    r = 8, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none",    # Supports any, but = "none" is optimized
    # [NEW] "unsloth" uses 30% less VRAM, fits 2x larger batch sizes!
    use_gradient_checkpointing = "unsloth", # True or "unsloth" for very long
context
    random_state = 3407,
    use_rslora = False,  # We support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)
```

```
Unsloth: Making `model.base_model.model.model` require gradients
```

Reasoning Effort

The `gpt-oss` models from OpenAI include a feature that allows users to adjust the model's "reasoning effort." This gives you control over the trade-off between the model's performance and its response speed (latency) which by the amount of token the model will use to think.

The `gpt-oss` models offer three distinct levels of reasoning effort you can choose from:

- **Low**: Optimized for tasks that need very fast responses and don't require complex, multi-step reasoning.
- **Medium**: A balance between performance and speed.
- **High**: Provides the strongest reasoning performance for tasks that require it, though this results in higher latency.

Data Prep

The `Startup Companies One-Line Pitches 2025` dataset will be utilized as our example. Startup Pitches Dataset (https://www.kaggle.com/datasets/pratyushpuri/startup-companies-one-line-pitches-2025) for fine-tuning. The goal of leveraging this dataset is to enable the model to learn how to expand concise startup pitches into detailed, structured business descriptions. Through this process, the model will develop reasoning capabilities and generate coherent, high-quality outputs that capture the key business context, technology, market potential, and innovation for each startup.

Data Loader

In [5]:
```
# newly added: Google Drive access
from google.colab import drive
drive.mount('/content/drive')
```

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

In [65]:
```
# newly added: Dataset
#Import Libraries
import pandas as pd
import numpy as np
from datasets import Dataset
#File path
csv_path = "/content/drive/MyDrive/Startup Companies One-Line Pitches 2025/sta
rtup_company_one_line_pitches.csv"
df= pd.read_csv(csv_path)
print(f"Shape of Datset: {df.shape}")

# Unique Industries
unique_industries = df['Industry'].nunique()
print("Number of unique Industries:", unique_industries)

# Unique Core Technologies
unique_technologies = df['Core_Technology'].nunique()
print("Number of unique Core Technologies:", unique_technologies)

# Function to sample 2 per group
def sample_per_category(df, group_cols, n=20):
    return (
        df.groupby(group_cols, group_keys=False)  # group by multiple columns
          .apply(lambda x: x.sample(min(len(x), n), random_state=42))  # sampl
e min(n, available)
          .reset_index(drop=True)
    )

# Sample 20 per Industry + Core_Technology
sampled_df = sample_per_category(df, ['Industry', 'Core_Technology'], n=2)

print(f"shape of sample dataframe: {sampled_df.shape}")

print("============================About Dataset=================================
=======")
sampled_df.head(5)
```

```
Shape of Datset: (3069, 15)
Number of unique Industries: 15
Number of unique Core Technologies: 15
shape of sample dataframe: (450, 15)
==========About Dataset==========
=

/tmp/ipython-input-2029217171.py:23: DeprecationWarning: DataFrameGroupBy.app
ly operated on the grouping columns. This behavior is deprecated, and in a fu
ture version of pandas the grouping columns will be excluded from the operati
on. Either pass `include_groups=False` to exclude the groupings or explicitly
select the grouping columns after groupby to silence this warning.
  .apply(lambda x: x.sample(min(len(x), n), random_state=42))  # sample min
(n, available)
```

Out[65]:

	Startup_ID	Company_Name	Industry	One_Line_Pitch	Founding_Year	Headquarters_Location
0	2719	Rose and Sons	AI & Machine Learning	Decentralized AI models for edge intelligence.	2011	West Jason, Jordan
1	2114	Cox Ltd	AI & Machine Learning	Hybrid AI approaches combining symbolic and ne...	2024	Patriciaburgh, Finland
2	2164	Stewart LLC	AI & Machine Learning	Deploying secure ML solutions with privacy-fir...	2015	Lake Kelsey, Bolivia
3	2487	Hale, Nelson and Brown	AI & Machine Learning	AI pipelines optimized for continuous delivery...	2024	Freemanmouth, American Samoa
4	178	Khan, Pena and Ruiz	AI & Machine Learning	AI in financial forecasting for smarter invest...	2013	Port Danielberg, Turkmenistan

In [66]:
```
# newly added: Convert a pre-sampled startup DataFrame into USloth prompt/answ
er format for unsloth/gpt-oss-20b.

from datasets import Dataset

def process_startup_df_to_usloth_format(sample_df):
    # Keep only relevant columns
    df = sample_df[['One_Line_Pitch', 'Company_Name', 'Industry', 'Core_Techno
logy', 'Market_Size_Billion_USD']].copy()

    def map_to_usloth_messages(example):
        # User prompt: minimal instruction with the pitch
        user_prompt = (f"Expand this startup idea into a detailed business des
cription:\n{example['One_Line_Pitch']}"
        )
        # Assistant response: dynamic template using placeholders and pronouns
        assistant_response = (
           f"\n{example['Company_Name']} operates in the {example['Industry']}
sector, focusing on '{example['One_Line_Pitch']}'. \n"
           f"They leverage {example['Core_Technology']} to address key challen
ges efficiently. \n"
           f"The market opportunity is around {example['Market_Size_Billion_US
D']} billion USD, offering strong growth potential. \n"
           f"Their technological approach gives them a competitive advantage,
enabling smarter, faster, and scalable solutions. \n"
           f"With this foundation, they are positioned to expand and make a si
gnificant impact within the {example['Industry']} landscape. \n"
        )
        return {
            "messages": [
                {"role": "user", "content": user_prompt},
                {"role": "assistant", "content": assistant_response, "channe
l": "final"}
            ]
        }

    # Convert DataFrame to HuggingFace Dataset and map messages
    ds = Dataset.from_pandas(df)
    ds = ds.map(map_to_usloth_messages)
    return ds

# Example usage:
dataset = process_startup_df_to_usloth_format(sampled_df)
```

In [67]:
```
def formatting_prompts_func(examples):
    convos = examples["messages"]
    texts = [tokenizer.apply_chat_template(convo, tokenize = False, add_genera
tion_prompt = False) for convo in convos]
    return { "text" : texts, }

#from datasets import load_dataset

#dataset = load_dataset("HuggingFaceH4/Multilingual-Thinking", split="train")
#dataset
```

To format our dataset, we will apply our version of the GPT OSS prompt

In [68]:
```
from unsloth.chat_templates import standardize_sharegpt
dataset = standardize_sharegpt(dataset)
dataset = dataset.map(formatting_prompts_func, batched = True,)
```

Let's take a look at the dataset, and check what the 1st example shows

In [69]:
```
print(dataset[0]['text'])
```

```
<|start|>system<|message|>You are ChatGPT, a large language model trained by
OpenAI.
Knowledge cutoff: 2024-06
Current date: 2025-11-16

Reasoning: medium

# Valid channels: analysis, commentary, final. Channel must be included for e
very message.
Calls to these tools must go to the commentary channel: 'functions'.<|end|><|
start|>user<|message|>Expand this startup idea into a detailed business descr
iption:
Decentralized AI models for edge intelligence.<|end|><|start|>assistant<|mess
age|>
Rose and Sons operates in the AI & Machine Learning sector, focusing on 'Dece
ntralized AI models for edge intelligence.'.
They leverage 5G to address key challenges efficiently.
The market opportunity is around 50.0 billion USD, offering strong growth pot
ential.
Their technological approach gives them a competitive advantage, enabling sma
rter, faster, and scalable solutions.
With this foundation, they are positioned to expand and make a significant im
pact within the AI & Machine Learning landscape.
<|return|>
```

What is unique about GPT-OSS is that it uses OpenAI Harmony (https://github.com/openai/harmony) format which support conversation structures, reasoning output, and tool calling.

Train the model

Now let's train our model. We do 60 steps to speed things up, but you can set `num_train_epochs=1` for a full run, and turn off `max_steps=None` .

In [106]:
```
from trl import SFTConfig, SFTTrainer
trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = dataset,
    args = SFTConfig(
        per_device_train_batch_size = 1,
        gradient_accumulation_steps = 16, #change it to 4 to 8 # large size ma
y cause out of memory
        warmup_steps = 5,
        num_train_epochs = 2, # Set this for 1 full training run.
        max_steps = 51, #[(total_smaple_size)*num_train_epochs]/(per_device_tr
ain_batch_size * gradient_accumulation_steps) # (405*2)/(1*16) # 51
        learning_rate = 2e-4,
        logging_steps = 1,
        optim = "adamw_8bit",
        weight_decay = 0.001,
        lr_scheduler_type = "linear",
        seed = 3407,
        output_dir = "outputs",
        report_to = "none", # Use TrackIO/WandB etc
    ),
)
```

```
Unsloth: Switching to float32 training since model cannot work with float16
```

We also use Unsloth's `train_on_completions` method to only train on the assistant outputs and ignore the loss on the user's inputs. This helps increase accuracy of finetunes and lower loss as well!

In [107]:
```
from unsloth.chat_templates import train_on_responses_only

#gpt_oss_kwargs = dict(instruction_part = "<|start|>user<|message|>", response
_part="<|start|>assistant<|channel|>final<|message|>")
# channel is an additional special token. since there is no addition special t
oken, we must remove it to avoid error
gpt_oss_kwargs = dict(
    instruction_part = "<|start|>user<|message|>",
    response_part    = "<|start|>assistant<|message|>",
)

trainer = train_on_responses_only(
    trainer,
    **gpt_oss_kwargs,
)
```

Let's verify masking the instruction part is done! Let's print the 100th row again.

In [108]:
```
tokenizer.decode(trainer.train_dataset[100]["input_ids"])
```

Out[108]:
```
"<|start|>system<|message|>You are ChatGPT, a large language model trained by
OpenAI.\nKnowledge cutoff: 2024-06\nCurrent date: 2025-11-16\n\nReasoning: me
dium\n\n# Valid channels: analysis, commentary, final. Channel must be includ
ed for every message.\nCalls to these tools must go to the commentary channe
l: 'functions'.<|end|><|start|>user<|message|>Expand this startup idea into a
detailed business description:\nThreat-hunting copilots enhancing SOC operati
ons.<|end|><|start|>assistant<|message|>\nSexton, Crosby and Evans operates i
n the Cybersecurity sector, focusing on 'Threat-hunting copilots enhancing SO
C operations.'. \nThey leverage Cloud Computing to address key challenges eff
iciently. \nThe market opportunity is around 10.0 billion USD, offering stron
g growth potential. \nTheir technological approach gives them a competitive a
dvantage, enabling smarter, faster, and scalable solutions. \nWith this found
ation, they are positioned to expand and make a significant impact within the
Cybersecurity landscape. \n<|return|>"
```

Now let's print the masked out example - you should see only the answer is present:

In [109]:
```
tokenizer.decode([tokenizer.pad_token_id if x == -100 else x for x in trainer.
train_dataset[100]["labels"]]).replace(tokenizer.pad_token, " ")
```

Out[109]:
```
"
\nSexton, Crosby and Evans operates in the Cybersecurity sector, focusing on
'Threat-hunting copilots enhancing SOC operations.'. \nThey leverage Cloud Co
mputing to address key challenges efficiently. \nThe market opportunity is ar
ound 10.0 billion USD, offering strong growth potential. \nTheir technologica
l approach gives them a competitive advantage, enabling smarter, faster, and
scalable solutions. \nWith this foundation, they are positioned to expand and
make a significant impact within the Cybersecurity landscape. \n<|return|>"
```

In [110]:
```
# @title Show current memory stats
gpu_stats = torch.cuda.get_device_properties(0)
start_gpu_memory = round(torch.cuda.max_memory_reserved() / 1024 / 1024 / 102
4, 3)
max_memory = round(gpu_stats.total_memory / 1024 / 1024 / 1024, 3)
print(f"GPU = {gpu_stats.name}. Max memory = {max_memory} GB.")
print(f"{start_gpu_memory} GB of memory reserved.")
```

```
GPU = Tesla T4. Max memory = 14.741 GB.
12.816 GB of memory reserved.
```

Let's train the model! To resume a training run, set `trainer.train(resume_from_checkpoint = True)`

In [111]:
```
trainer_stats = trainer.train()
```

```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs used = 1
   \\   /|    Num examples = 450 | Num Epochs = 2 | Total steps = 51
O^O/ \_/ \    Batch size per device = 1 | Gradient accumulation steps = 16
\        /    Data Parallel GPUs = 1 | Total batch size (1 x 16 x 1) = 16
 "-____-"     Trainable parameters = 3,981,312 of 20,918,738,496 (0.02% train
ed)
```

[51/51 46:02, Epoch 1/2]

Step	Training Loss
1	0.219400
2	0.226700
3	0.219800
4	0.218600
5	0.229100
6	0.207100
7	0.235800
8	0.224000
9	0.221500
10	0.240100
11	0.253800
12	0.246400
13	0.240500
14	0.226100
15	0.246500
16	0.256600
17	0.254400
18	0.228900
19	0.226300
20	0.224100
21	0.247100
22	0.233900
23	0.232700
24	0.238400
25	0.248700
26	0.221800
27	0.247200
28	0.231100
29	0.267800
30	0.239200
31	0.240600
32	0.216900
33	0.208600
34	0.208100
35	0.223700

Step	Training Loss
36	0.212700
37	0.218400
38	0.232600
39	0.208700
40	0.252600
41	0.223100
42	0.242500
43	0.236700
44	0.239200
45	0.209700
46	0.205300
47	0.208500
48	0.207300
49	0.200500
50	0.245300
51	0.236500

In [112]:
```
# @title Show final memory and time stats
used_memory = round(torch.cuda.max_memory_reserved() / 1024 / 1024 / 1024, 3)
used_memory_for_lora = round(used_memory - start_gpu_memory, 3)
used_percentage = round(used_memory / max_memory * 100, 3)
lora_percentage = round(used_memory_for_lora / max_memory * 100, 3)
print(f"{trainer_stats.metrics['train_runtime']} seconds used for training.")
print(
    f"{round(trainer_stats.metrics['train_runtime']/60, 2)} minutes used for t
raining."
)
print(f"Peak reserved memory = {used_memory} GB.")
print(f"Peak reserved memory for training = {used_memory_for_lora} GB.")
print(f"Peak reserved memory % of max memory = {used_percentage} %.")
print(f"Peak reserved memory for training % of max memory = {lora_percentage}
%.")
```

```
2819.0174 seconds used for training.
46.98 minutes used for training.
Peak reserved memory = 12.816 GB.
Peak reserved memory for training = 0.0 GB.
Peak reserved memory % of max memory = 86.941 %.
Peak reserved memory for training % of max memory = 0.0 %.
```

Inference

Let's run the model! You can change the instruction and input - leave the output blank!

In [113]:
```
messages = [
    #{"role": "system", "content": "reasoning language: French\n\nYou are a helpful assistant that can solve mathematical problems."},
    {"role": "user", "content": "Deploying secure ML solutions with privacy-first design."},
]
inputs = tokenizer.apply_chat_template(
    messages,
    add_generation_prompt = True,
    return_tensors = "pt",
    return_dict = True,
    reasoning_effort = "medium",
).to("cuda")
from transformers import TextStreamer
_ = model.generate(**inputs, max_new_tokens = 150, streamer = TextStreamer(tokenizer))
```

```
<|start|>system<|message|>You are ChatGPT, a large language model trained by OpenAI.
Knowledge cutoff: 2024-06
Current date: 2025-11-16

Reasoning: medium

# Valid channels: analysis, commentary, final. Channel must be included for every message.
Calls to these tools must go to the commentary channel: 'functions'.<|end|><|start|>user<|message|>Deploying secure ML solutions with privacy-first design.<|end|><|start|>assistant<|channel|>elksworth-brockwell.net Adkins Group, 30 West 8th Street, Plano, Texas, 75002.
Williams operates in the Gaming sector at 30 West 8th Street, Plano, Texas.
They focus on Quantum Computing, leveraging Cloud Security to drive growth.
Their technological approach gives them a competitive advantage, enabling efficient, scalable, and secure solutions. <|return|>
```

In [116]:
```
# Generate output and store as tensor
outputs = model.generate(
    **inputs,
    max_new_tokens=150,
)# Decode to get the text string
generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)

# Print in a chat-style format
print("\n--- Chat Output ---\n")
print(f"User: {messages[0]['content']}")
print(f"\n Assistant:\n{generated_text}\n")
```

```
--- Chat Output ---

User: Deploying secure ML solutions with privacy-first design.

 Assistant:
systemYou are ChatGPT, a large language model trained by OpenAI.
Knowledge cutoff: 2024-06
Current date: 2025-11-16

Reasoning: medium

# Valid channels: analysis, commentary, final. Channel must be included for e
very message.
Calls to these tools must go to the commentary channel: 'functions'.userDeplo
ying secure ML solutions with privacy-first design.assistantayala-hernandez.n
et responds as David Ortiz.
Ibrahim Davis serves as the Technology Lead at Ayala-Hernandez.net.
He focuses on Artificial Intelligence, leveraging Cloud Computing to address
key challenges efficiently.
His approach ensures efficient, scalable, and secure solutions, embracing cut
ting-edge technologies to drive growth and innovation consistently.
The impact of his work spans across the Home and Building sectors, showcasing
a strong influence through thoughtful leadership and technological expertise.
Giovanni Ramirez holds the position of Cybersecurity Manager at Ayala-Hernand
ez.net.
His background in Computer Science equips him to tackle complex challenges wi
th a focus on Emerging Technologies, Cybersecurity, and Cloud Computing.
Within the Home and Building landscape, his contributions stand
```

Saving, loading finetuned models

To save the final model as LoRA adapters, either use Huggingface's `push_to_hub` for an online save or `save_pretrained` for a local save.

[NOTE] Currently finetunes can only be loaded via Unsloth in the meantime - we're working on vLLM and GGUF exporting!

In [117]:
```
model.save_pretrained("finetuned_model_startup_2025")
# model.push_to_hub("hf_username/finetuned_model", token = "hf_...") # Save to
HF
```

To run the finetuned model, you can do the below after setting `if False` to `if True` in a new instance.

In [119]:
```
if False:
    from unsloth import FastLanguageModel
    model, tokenizer = FastLanguageModel.from_pretrained(
        model_name = "finetuned_model_startup_2025", # YOUR MODEL YOU USED FOR
TRAINING
        max_seq_length = 1024,
        dtype = None,
        load_in_4bit = True,
    )

messages = [
    #{"role": "system", "content": "You are a helpful assistant. Generate a st
ructured startup business description in a professional style. "},
    {"role": "user", "content": "Deploying secure ML solutions with privacy-fi
rst design."},
]
inputs = tokenizer.apply_chat_template(
    messages,
    add_generation_prompt = True,
    return_tensors = "pt",
    return_dict = True,
    reasoning_effort = "medium",
).to("cuda")
from transformers import TextStreamer
_ = model.generate(**inputs, max_new_tokens = 150, streamer = TextStreamer(tok
enizer))
```

```
<|start|>system<|message|>You are ChatGPT, a large language model trained by
OpenAI.
Knowledge cutoff: 2024-06
Current date: 2025-11-16

Reasoning: medium

# Valid channels: analysis, commentary, final. Channel must be included for e
very message.
Calls to these tools must go to the commentary channel: 'functions'.<|end|><|
start|>user<|message|>Deploying secure ML solutions with privacy-first desig
n.<|end|><|start|>assistant<|channel|>achton-Harris-Lambert operated 26.5 Squ
are Meters' Touchstone Center' in Orlando, Florida. They focus on Machine Lea
rning, Virtual Reality, and Cybersecurity. The Global Tech operates 26.5 Squa
re Meters, focusing on Cloud Computing, Artificial Intelligence, and Cybersec
urity. They leverage Cloud, GoPivots into Cybersecurity, leveraging Artificia
l Intelligence for efficient, scalable solutions. The Growth Frontiers operat
es 26.5 Square Meters, focusing on Mobile Platforms, Cloud Computing, and Art
ificial Intelligence. They leverage Big Data to drive efficiencies in Cyberse
curity, focusing on Cloud Computing, Artificial Intelligence, and Cybersecuri
ty.<|return|>
```

In [121]:
```
# Generate output and store as tensor
outputs = model.generate(
    **inputs,
    max_new_tokens=150,
)# Decode to get the text string
generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)

# Print in a chat-style format
print("\n--- Chat Output ---\n")
print(f"User: {messages[0]['content']}")
print(f"\n Assistant:\n{generated_text}\n")
```

```
--- Chat Output ---

User: Deploying secure ML solutions with privacy-first design.

 Assistant:
systemYou are ChatGPT, a large language model trained by OpenAI.
Knowledge cutoff: 2024-06
Current date: 2025-11-16

Reasoning: medium

# Valid channels: analysis, commentary, final. Channel must be included for e
very message.
Calls to these tools must go to the commentary channel: 'functions'.userDeplo
ying secure ML solutions with privacy-first design.assistanteliseo_kaufman re
plied 2025-11-16 21:54:18
Carreon Group operates in the Cybersecurity sector, focusing on 'Deploying se
cure ML solutions with privacy-first design.'. They leverage Cloud Computing
to address key challenges efficiently. The market opportunity is around 30.0
billion USD, offering strong growth potential. Their technological approach g
ives them a competitive advantage, enabling smarter, faster, and scalable sol
utions.
```

Saving to float16 for VLLM or mxfp4

We also support saving to `float16` or `mxfp4` directly. Select `merged_16bit` for float16. Use `push_to_hub_merged` to upload to your Hugging Face account! You can go to https://huggingface.co/settings/tokens (https://huggingface.co/settings/tokens) for your personal tokens.

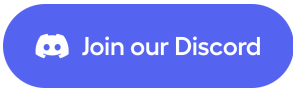
In [ ]:
```
# Merge and push to hub in mxfp4 4bit format
if False:
    model.save_pretrained_merged("finetuned_model", tokenizer, save_method =
"mxfp4")
if False: model.push_to_hub_merged("repo_id/repo_name", tokenizer, token = "h
f...", save_method = "mxfp4")

# Merge and push to hub in 16bit
if False:
    model.save_pretrained_merged("finetuned_model", tokenizer, save_method =
"merged_16bit")
if False: # Pushing to HF Hub
    model.push_to_hub_merged("hf/gpt-oss-finetune", tokenizer, save_method =
"merged_16bit", token = "")
```

And we're done! If you have any questions on Unsloth, we have a Discord (https://discord.gg/unsloth) channel! If you find any bugs or want to keep updated with the latest LLM stuff, or need help, join projects etc, feel free to join our Discord!

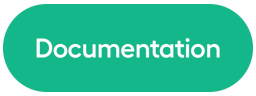
Some other links:

1. Train your own reasoning model - Llama GRPO notebook Free Colab (https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Llama3.1_(8B)-GRPO.ipynb)
2. Saving finetunes to Ollama. Free notebook (https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Llama3_(8B)-Ollama.ipynb)
3. Llama 3.2 Vision finetuning - Radiography use case. Free Colab (https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Llama3.2_(11B)-Vision.ipynb)
4. See notebooks for DPO, ORPO, Continued pretraining, conversational finetuning and more on our documentation (https://docs.unsloth.ai/get-started/unsloth-notebooks)!

(https://unsloth.ai)



(https://discord.gg/unsloth)



(https://docs.unsloth.ai/) Join Discord if you need help + ⭐ *Star us on Github (https://github.com/unslothai/unsloth)* ⭐

This notebook and all Unsloth notebooks are licensed LGPL-3.0 (https://github.com/unslothai/notebooks?tab=LGPL-3.0-1-ov-file#readme).