Fine-tuning Model Qwen2 for Indonesian Food Recipes

This notebook fine-tunes the Qwen2 Instruct model on Indonesian food recipes using LoRA.

In [1]:
```
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer, TrainingArguments, Tr
import pandas as pd
from datasets import Dataset
import os
from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training, TaskT
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import display, HTML
```

```
D:\AppLabs\chef-ai\venv\Lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress no
t found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.i
o/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

Configuration

In [2]:
```
# MODEL_NAME = "Qwen/Qwen2.5-1.5B-Instruct"
MODEL_NAME = "Qwen/Qwen2-0.5B-Instruct"
OUTPUT_DIR = "./IFMF-Qwen2-0.5B-Instruct-full"
CSV_FILE = "./dataset_all/Indonesian_Food_Recipes_full.csv"

device = "cuda" if torch.cuda.is_available() else "cpu"
print(f"Using device: {device}")
```

```
Using device: cuda
```

Load Model and Tokenizer

In [3]:
```
print(f"Loading model and tokenizer from {MODEL_NAME}...")
model = AutoModelForCausalLM.from_pretrained(
    MODEL_NAME,
    torch_dtype="auto",
    device_map="auto",  # Use auto device mapping for better memory management
    # low_cpu_mem_usage=True,
    # load_in_8bit=True,  # Load model in 8-bit precision to reduce memory usage
    use_cache=False  # Disable KV cache for compatibility with gradient checkpointi
)
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)

# Set pad token if not set
if tokenizer.pad_token_id is None:
    tokenizer.pad_token_id = tokenizer.eos_token_id
```

```
Loading model and tokenizer from Qwen/Qwen2-0.5B-Instruct...
Sliding Window Attention is enabled but not implemented for `sdpa`; unexpected resul
ts may be encountered.
```

Prepare Model for LoRA Fine-tuning

In [4]:
```python
# Prepare the model for training with LoRA
print("Preparing model for training with LoRA...")
model = prepare_model_for_kbit_training(model)
base_model_config = model.config.to_dict()
if "model_type" not in base_model_config:
    base_model_config["model_type"] = "qwen"  # Set model_type for Qwen model

# Create output directory if it doesn't exist
os.makedirs(OUTPUT_DIR, exist_ok=True)

# Save the updated config
with open(os.path.join(OUTPUT_DIR, "config.json"), "w") as f:
    import json
    json.dump(base_model_config, f)

# Define LoRA configuration
lora_config = LoraConfig(
    r=16,  # Rank
    lora_alpha=32,
    target_modules=["q_proj", "v_proj", "k_proj", "o_proj", "gate_proj", "up_proj",
    lora_dropout=0.05,
    bias="none",
    task_type=TaskType.CAUSAL_LM
)

# Apply LoRA to the model
model = get_peft_model(model, lora_config)
print("LoRA adapters added to the model")
model.print_trainable_parameters()
```

```
Preparing model for training with LoRA...
LoRA adapters added to the model
trainable params: 8,798,208 || all params: 502,830,976 || trainable%: 1.7497
```

Load and Prepare Dataset

In [5]:
```python
print("Loading dataset...")
dataset_path = CSV_FILE

if not os.path.exists(dataset_path):
    raise FileNotFoundError(f"Dataset file not found at {dataset_path}. Please ensu

df = pd.read_csv(dataset_path)

# Print dataset columns
print("Dataset columns:")
print(df.columns)
```

```
print(f"Total data in dataset: {len(df)}")

# Display a sample row
display(df.head(10))
```

```
Loading dataset...
Dataset columns:
Index(['Title', 'Ingredients', 'Steps', 'Loves', 'URL', 'Category',
       'Title Cleaned', 'Total Ingredients', 'Ingredients Cleaned',
       'Total Steps'],
      dtype='object')
Total data in dataset: 14945
```

	Title	Ingredients	Steps	Loves	URL	Category
0	Ayam Woku Manado	1 Ekor Ayam Kampung (potong 12)--2 Buah Jeruk ...	1) Cuci bersih ayam dan tiriskan. Lalu peras j...	1	https://cookpad.com/id/resep/4473027-ayam-woku...	ayam
1	Ayam goreng tulang lunak	1 kg ayam (dipotong sesuai selera jangan kecil...	1) Haluskan bumbu2nya (BaPut, ketumbar, kemiri...	1	https://cookpad.com/id/resep/4471956-ayam-gore...	ayam
2	Ayam cabai kawin	1/4 kg ayam--3 buah cabai hijau besar--7 buah ...	1) Panaskan minyak di dalam wajan. Setelah min...	2	https://cookpad.com/id/resep/4473057-ayam-caba...	ayam
3	Ayam Geprek	250 gr daging ayam (saya pakai fillet)--Secuku...	1) Goreng ayam seperti ayam krispi\n2) Ulek se...	10	https://cookpad.com/id/resep/4473023-ayam-geprek	ayam
4	Minyak Ayam	400 gr kulit ayam & lemaknya--8 siung bawang p...	1) Cuci bersih kulit ayam. Sisihkan\n2) Ambil ...	4	https://cookpad.com/id/resep/4427438-minyak-ayam	ayam
5	Nasi Bakar Ayam	1 piring nasi--1/4 fillet ayam, potong kotak, ...	1) Tumis bumbu halus, masukkan daun salam, ser...	1	https://cookpad.com/id/resep/4472552-nasi-baka...	ayam
6	Ayam Saus Hintalu Jaruk	1/2 Ekor ayam--2 Butir Hintalu Jaruk--1 Buah C...	1) Potong ayam menjadi kotak-kotak ukuran seda...	0	https://cookpad.com/id/resep/4437475-ayam-saus...	ayam
7	Ayam saos teriyaki Lada Hitam	Ayam bagian dada dan tulang--1 buah	1) Cara Buat :\n1. Sediakan teplon or wajan be...	0	https://cookpad.com/id/resep/4472877-ayam-saos...	ayam

	Title	Ingredients	Steps	Loves	URL	Category
		bawang bom...				
8	Steak ayam	300 gr dada ayam fillet--1 sdm air jeruk nipis...	1) Cuci bersih ayam, iris tipis melebar, renda...	6	https://cookpad.com/id/resep/4472822-steak-ayam	ayam
9	Ayam Saos Asam Manis Simple	1/4 kg Ayam bagian dada fillet (Potong dadu)--...	1) Lumuri ayam yg sdh dipotong dadu dgn garam ...	6	https://cookpad.com/id/resep/4472901-ayam-saos...	ayam

In [6]:
```
def prepare_data(row):
    # Format the recipe data into a chat format
    system_msg = "Kamu adalah Chef Indonesia yang ahli dalam masakan tradisional. T
    user_msg = f"Tolong ajarkan saya resep lengkap untuk membuat {row['Title']}. Sa
    assistant_msg = f"""Saya akan membantu Anda membuat {row['Title']}.

Berikut adalah bahan-bahan yang diperlukan:
{row['Ingredients']}

Langkah-langkah pembuatan:
{row['Steps']}

Tips:
- Pastikan semua bahan sudah disiapkan sebelum mulai memasak
- Ikuti langkah-langkah dengan teliti untuk hasil terbaik
- Sesuaikan tingkat kepedasan dan rasa sesuai selera"""

    messages = [
        {"role": "system", "content": system_msg},
        {"role": "user", "content": user_msg},
        {"role": "assistant", "content": assistant_msg}
    ]

    # Apply chat template
    return {"text": tokenizer.apply_chat_template(messages, tokenize=False)}

# Convert DataFrame to Dataset
dataset = Dataset.from_pandas(df)
dataset = dataset.map(prepare_data)
print(f"Total examples after preparation: {len(dataset)}")

# Show a sample of prepared data
print("\nSample of prepared data:")
print(dataset[0]['text'][:500] + "...")
```

```
Map: 100%|████████████████████████████████████████████████████████| 14945/1494
5 [00:02<00:00, 5196.64 examples/s]
Total examples after preparation: 14945

Sample of prepared data:
<|im_start|>system
Kamu adalah Chef Indonesia yang ahli dalam masakan tradisional. Tugasmu adalah membe
rikan resep lengkap dengan bahan-bahan dan langkah memasak yang detail dan mudah dii
kuti.<|im_end|>
<|im_start|>user
Tolong ajarkan saya resep lengkap untuk membuat Ayam Woku Manado. Saya ingin mengeta
hui bahan-bahan dan langkah-langkahnya secara detail.<|im_end|>
<|im_start|>assistant
Saya akan membantu Anda membuat Ayam Woku Manado.

Berikut adalah bahan-bahan yang diperlukan:
1 Ekor Ayam Kam...
```

In [7]:
```python
def tokenize_function(examples):
    return tokenizer(
        examples["text"],
        padding="max_length",
        truncation=True,
        max_length=512*2  # Reduced from 512 to save memory
    )

# Tokenize dataset
tokenized_dataset = dataset.map(
    tokenize_function,
    batched=True,
    remove_columns=dataset.column_names
)
print(f"Total examples after tokenization: {len(tokenized_dataset)}")
```

```
Map: 100%|████████████████████████████████████████████████████████| 14945/1494
5 [00:06<00:00, 2214.33 examples/s]
Total examples after tokenization: 14945
```

Training Setup

In [8]:
```python
# Create data collator for language modeling
data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,
    mlm=False  # We're doing causal language modeling, not masked language modeling
)

# Custom callback to track and plot training metrics
from transformers.trainer_callback import TrainerCallback

class PlotLossCallback(TrainerCallback):
    def __init__(self):
        self.training_loss = []
        self.learning_rates = []
        self.steps = []
```

```
    def on_log(self, args, state, control, logs=None, **kwargs):
        if logs is not None and "loss" in logs:
            self.training_loss.append(logs["loss"])
            self.steps.append(state.global_step)
            if "learning_rate" in logs:
                self.learning_rates.append(logs["learning_rate"])

            # Plot every 10 logs
            if len(self.steps) % 10 == 0:
                self.plot_metrics()

    def plot_metrics(self):
        plt.figure(figsize=(15, 5))

        # Plot training loss
        plt.subplot(1, 2, 1)
        plt.plot(self.steps, self.training_loss)
        plt.xlabel('Steps')
        plt.ylabel('Training Loss')
        plt.title('Training Loss vs Steps')
        plt.grid(True)

        # Plot learning rate
        if self.learning_rates:
            plt.subplot(1, 2, 2)
            plt.plot(self.steps, self.learning_rates)
            plt.xlabel('Steps')
            plt.ylabel('Learning Rate')
            plt.title('Learning Rate vs Steps')
            plt.grid(True)

        plt.tight_layout()
        plt.show()

plot_callback = PlotLossCallback()
```

In [9]:

```
# Define training arguments
training_args = TrainingArguments(
    output_dir=OUTPUT_DIR,
    num_train_epochs=3,
    per_device_train_batch_size=1,  # Reduced batch size to save memory
    gradient_accumulation_steps=4,  # Accumulate gradients to simulate larger batch
    save_steps=500,
    save_total_limit=2,
    logging_dir="./logs",
    logging_steps=100,
    learning_rate=2e-5,
    weight_decay=0.01,
    fp16=True if torch.cuda.is_available() else False,
    # prediction_loss_only=False,  # Only return loss during training
    # gradient_checkpointing=True,  # Enable gradient checkpointing to save memory
    # Adding label_smoothing_factor to prevent overfitting and potential numerical
    label_smoothing_factor=0.1
)
```

```
# Initialize Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    data_collator=data_collator,  # Use the data collator instead of tokenizer
    # label_names=["input_ids", "attention_mask"],  # Explicitly provide label_name
    callbacks=[plot_callback]
)
```

```
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hi
des base models input arguments, if label_names is not given, label_names can't be s
et automatically within `Trainer`. Note that empty label_names list will be used ins
tead.
```

Start Training

In [10]:
```
print("Starting training...")
trainer.train()
```

```
Starting training...
```

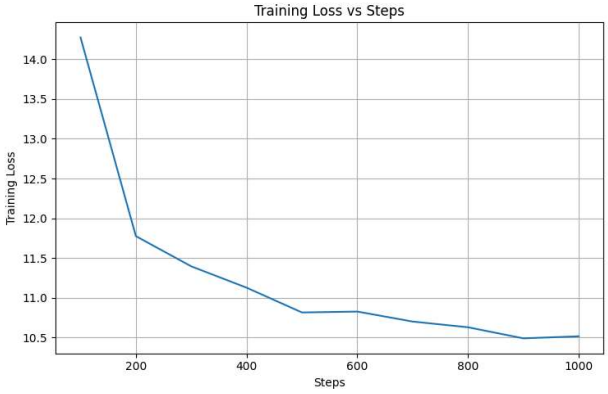
[11208/11208 4:21:42, Epoch 2/3]

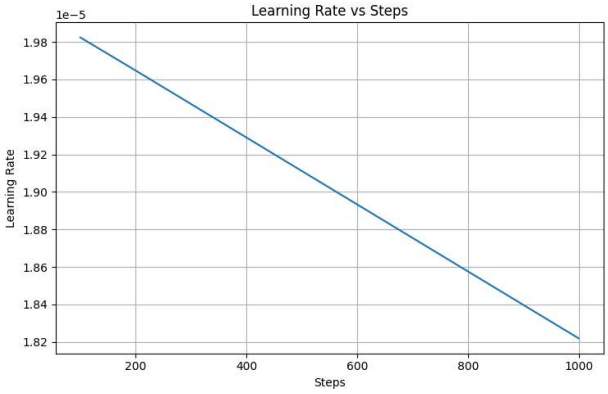
Step	Training Loss
100	14.274500
200	11.774900
300	11.395200
400	11.126500
500	10.814900
600	10.826000
700	10.700300
800	10.628400
900	10.489000
1000	10.515100
1100	10.391100
1200	10.410000
1300	10.334100
1400	10.306800
1500	10.359800
1600	10.309100
1700	10.222100
1800	10.272000
1900	10.331500
2000	10.094300
2100	10.172800
2200	10.250800
2300	10.173100
2400	10.271600
2500	10.284500
2600	10.136300
2700	10.062400
2800	10.126600
2900	10.069500
3000	10.122300

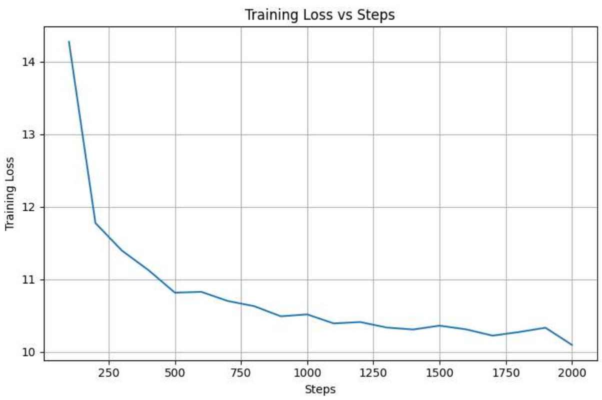
Step	Training Loss
3100	10.007700
3200	10.013700
3300	9.966800
3400	10.096800
3500	10.080500
3600	9.935500
3700	9.979900
3800	9.867500
3900	9.868100
4000	9.957300
4100	9.903000
4200	9.956900
4300	9.997900
4400	9.958500
4500	9.948600
4600	9.914900
4700	9.974500
4800	9.900100
4900	9.873500
5000	9.902900
5100	9.902300
5200	9.945500
5300	9.984600
5400	9.848300
5500	9.887600
5600	9.961700
5700	9.831100
5800	9.799500
5900	9.842200
6000	9.940800

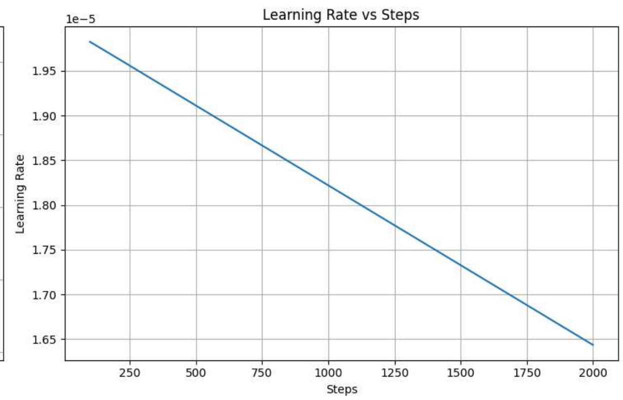
Step	Training Loss
6100	9.849500
6200	9.819000
6300	9.883600
6400	9.855100
6500	9.864900
6600	9.759200
6700	9.864100
6800	9.867700
6900	9.783200
7000	9.858000
7100	9.836400
7200	9.825800
7300	9.727600
7400	9.796500
7500	9.742100
7600	9.832800
7700	9.777500
7800	9.671300
7900	9.774600
8000	9.788300
8100	9.814400
8200	9.762000
8300	9.750300
8400	9.693200
8500	9.750200
8600	9.758400
8700	9.718400
8800	9.739500
8900	9.715600
9000	9.636600

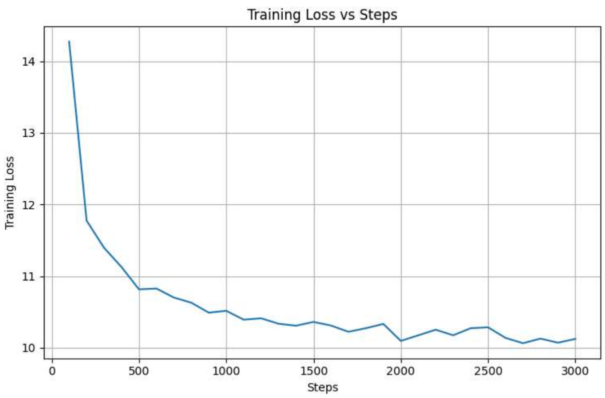
Step	Training Loss
9100	9.789900
9200	9.844800
9300	9.728800
9400	9.720100
9500	9.736100
9600	9.685500
9700	9.737200
9800	9.719200
9900	9.730300
10000	9.764900
10100	9.846600
10200	9.730900
10300	9.670500
10400	9.677000
10500	9.724900
10600	9.764600
10700	9.780800
10800	9.788300
10900	9.743200
11000	9.755300
11100	9.741400
11200	9.799000

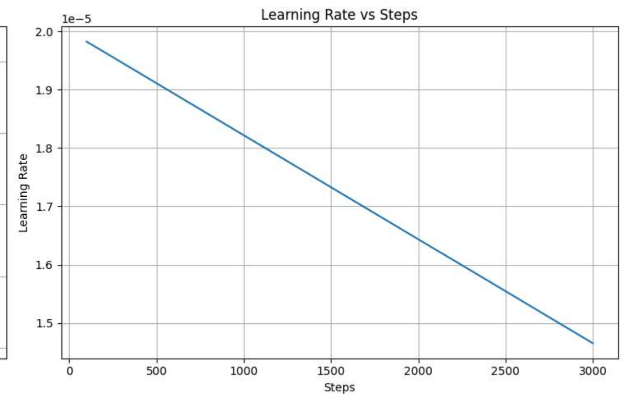


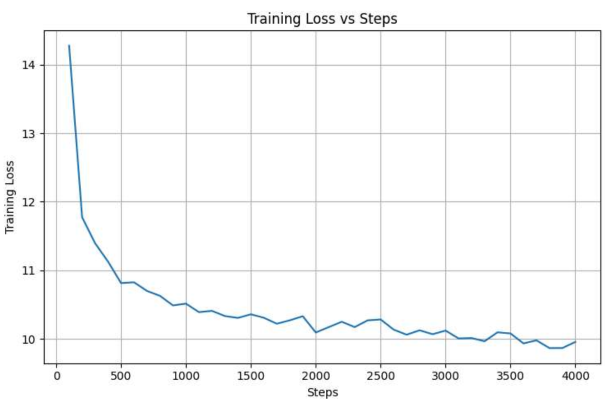


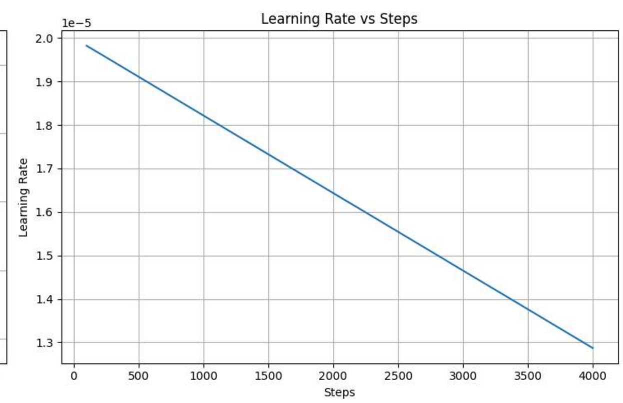

Training Loss vs Steps
Training Loss
Steps

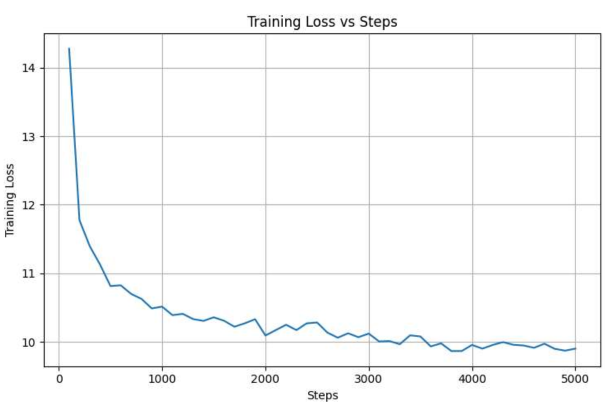

Learning Rate vs Steps
1e−5
Learning Rate
Steps

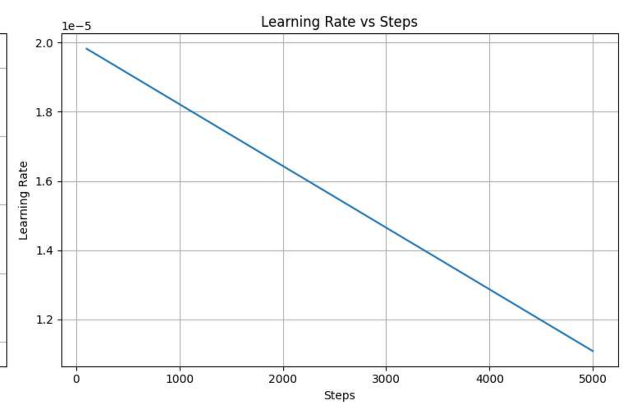

Training Loss vs Steps
Training Loss
Steps

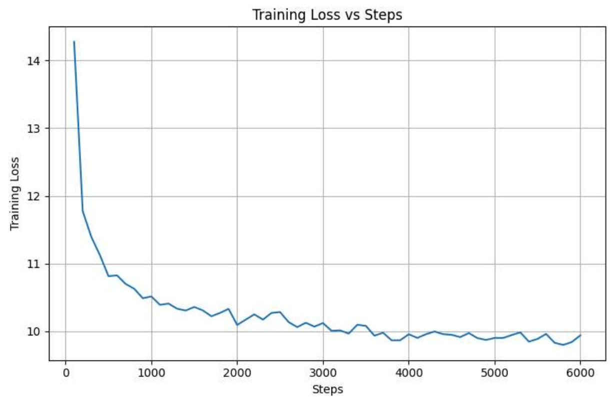

Learning Rate vs Steps
1e−5
Learning Rate
Steps

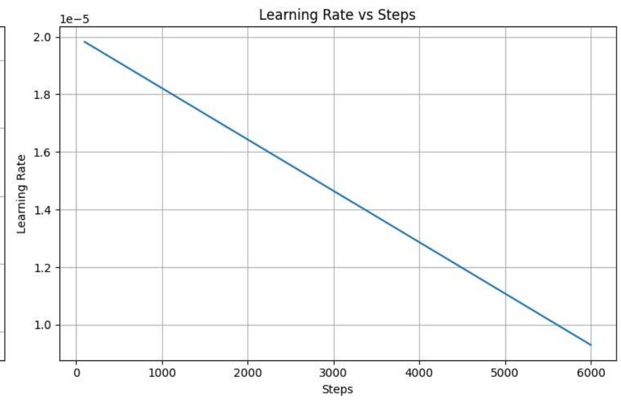

Training Loss vs Steps
Training Loss
Steps

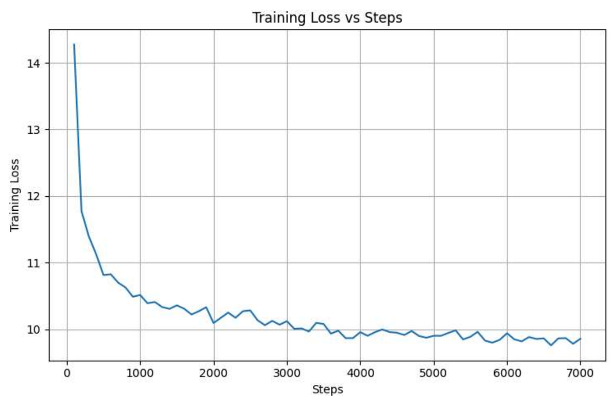

Learning Rate vs Steps
1e−5
Learning Rate
Steps

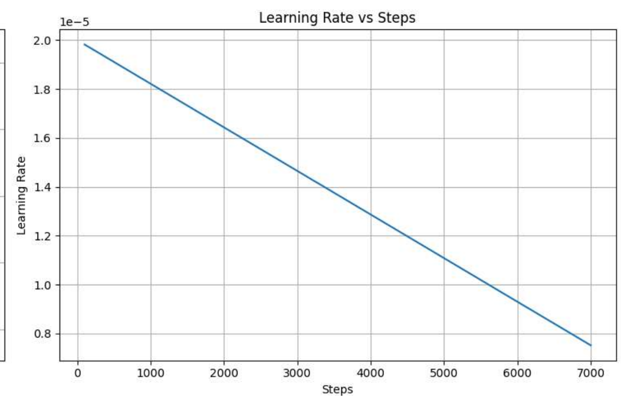

Training Loss vs Steps
Training Loss
Steps

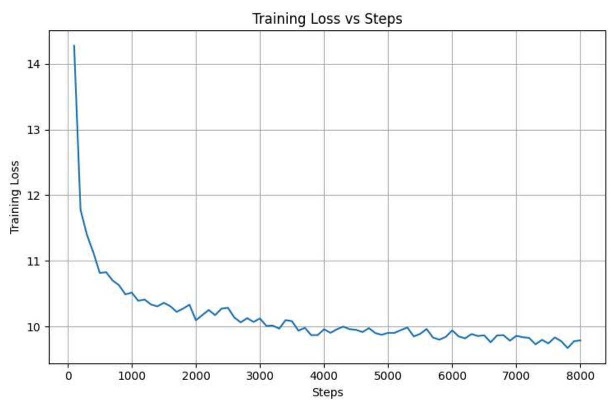

Learning Rate vs Steps
1e−5
Learning Rate
Steps

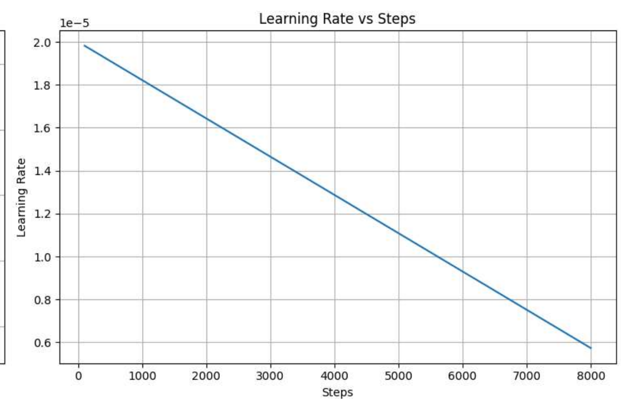

Training Loss vs Steps
Training Loss
Steps
10 11 12 13 14
0 1000 2000 3000 4000 5000 6000

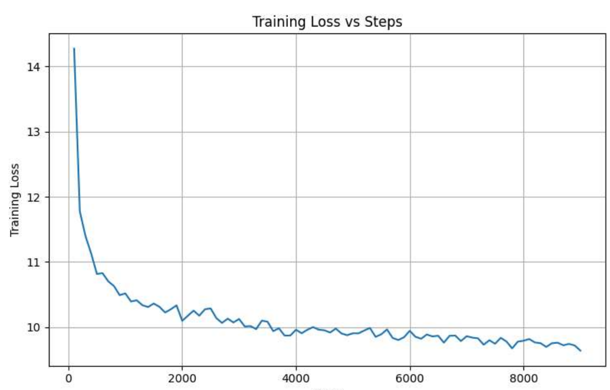

Learning Rate vs Steps
1e−5
Learning Rate
Steps
1.0 1.2 1.4 1.6 1.8 2.0
0 1000 2000 3000 4000 5000 6000

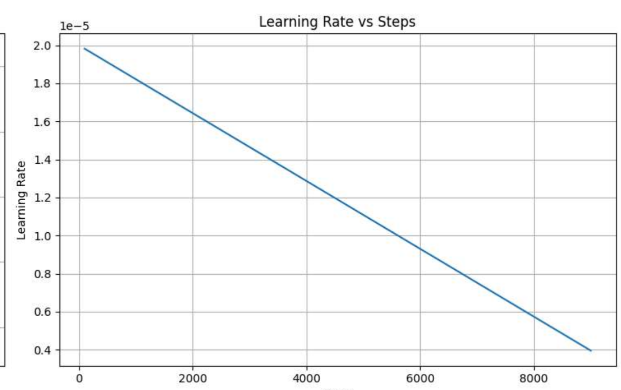

Training Loss vs Steps
Training Loss
Steps
10 11 12 13 14
0 1000 2000 3000 4000 5000 6000 7000

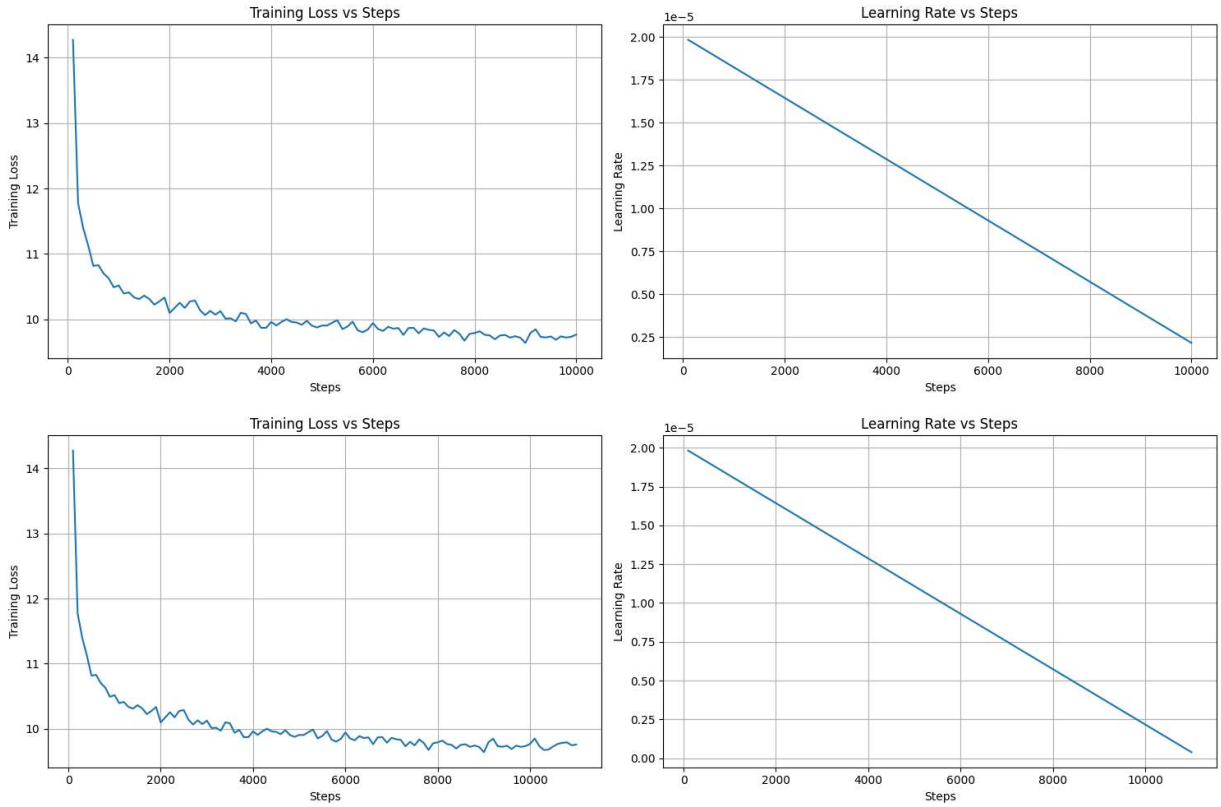

Learning Rate vs Steps
1e−5
Learning Rate
Steps
0.8 1.0 1.2 1.4 1.6 1.8 2.0
0 1000 2000 3000 4000 5000 6000 7000


Training Loss vs Steps
Training Loss
Steps
10 11 12 13 14
0 1000 2000 3000 4000 5000 6000 7000 8000


Learning Rate vs Steps
1e−5
Learning Rate
Steps
0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
0 1000 2000 3000 4000 5000 6000 7000 8000


Training Loss vs Steps
Training Loss
Steps
10 11 12 13 14
0 2000 4000 6000 8000


Learning Rate vs Steps
1e−5
Learning Rate
Steps
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
0 2000 4000 6000 8000



Out[10]:
```
TrainOutput(global_step=11208, training_loss=10.029430623568441, metrics={'train_r
untime': 15704.6978, 'train_samples_per_second': 2.855, 'train_steps_per_second':
0.714, 'total_flos': 1.0099217965252608e+17, 'train_loss': 10.029430623568441, 'ep
och': 2.9993977919036467})
```

Final Training Metrics

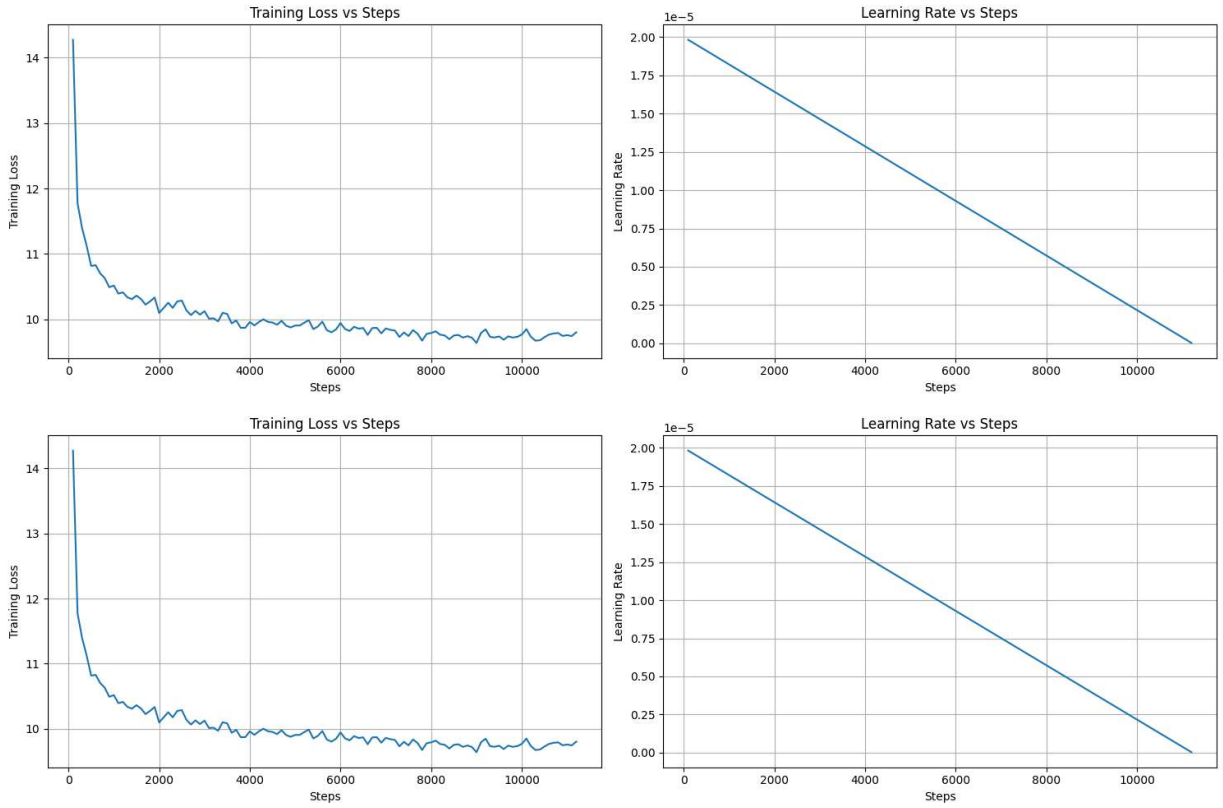
In [11]:
```python
# Plot final training metrics
plot_callback.plot_metrics()

# Save the plots
plt.figure(figsize=(15, 5))

# Plot training loss
plt.subplot(1, 2, 1)
plt.plot(plot_callback.steps, plot_callback.training_loss)
plt.xlabel('Steps')
plt.ylabel('Training Loss')
plt.title('Training Loss vs Steps')
plt.grid(True)

# Plot learning rate
if plot_callback.learning_rates:
    plt.subplot(1, 2, 2)
    plt.plot(plot_callback.steps, plot_callback.learning_rates)
    plt.xlabel('Steps')
    plt.ylabel('Learning Rate')
    plt.title('Learning Rate vs Steps')
    plt.grid(True)
```

```
plt.tight_layout()
plt.savefig(os.path.join(OUTPUT_DIR, "training_metrics.png"))
```



Save the Model

In [12]:
```
print("Saving model...")
trainer.save_model(OUTPUT_DIR)
print("Training completed!")
```

```
Saving model...
Training completed!
```

Test the Fine-tuned Model

In [13]:
```
# Load the fine-tuned model for testing
from peft import PeftModel, PeftConfig
import torch

# Load the base model
base_model = AutoModelForCausalLM.from_pretrained(
    MODEL_NAME,
    torch_dtype=torch.float16,  # Explicitly set dtype
    device_map="auto"
)

# Load the LoRA adapter
model = PeftModel.from_pretrained(base_model, OUTPUT_DIR)

# Ensure model is in evaluation mode
```

```
model.eval()

# Test with a sample prompt
test_prompt = "Tolong ajarkan saya resep lengkap untuk membuat Ayam Woku Manado. Sa

messages = [
    {"role": "system", "content": "Kamu adalah Chef Indonesia yang ahli dalam masak
    {"role": "user", "content": test_prompt}
]

input_text = tokenizer.apply_chat_template(messages, tokenize=False)
# Make sure tokenizer output is on the same device as the model
inputs = tokenizer(input_text, return_tensors="pt")
# Move inputs to the same device as model
device = next(model.parameters()).device
inputs = {k: v.to(device) for k, v in inputs.items()}

# Generate response
with torch.no_grad():
    outputs = model.generate(
        **inputs,
        max_new_tokens=1024,
        temperature=0.7,
        top_p=0.9,
        do_sample=True
    )

response = tokenizer.decode(outputs[0], skip_special_tokens=True)
print("Model response:")
print(response)
```

```
Model response:
system
Kamu adalah Chef Indonesia yang ahli dalam masakan tradisional. Tugasmu adalah membe
rikan resep lengkap dengan bahan-bahan dan langkah memasak yang detail dan mudah dii
kuti.
user
Tolong ajarkan saya resep lengkap untuk membuat Ayam Woku Manado. Saya ingin mengeta
hui bahan-bahan dan langkah-langkahnya secara detail.
assistant
Saya akan membantu Anda membuat Ayam Woku Manado.

Berikut adalah bahan-bahan yang diperlukan:
1/2 kg ayam--Bumbu halus :--3 siung bawang merah--3 siung bawang putih--5 buah cabe
rawit--3 buah tomat merah--2 sdm gula pasir--1 sdt lada bubuk--secukupnya Kecap mani
s--

Langkah-langkah pembuatan:
1) Rebus ayam sampai empuk, tiriskan.
2) Haluskan bumbu halus. Lalu tumis bumbu sampai harum.
3) Masukkan ayam ke dalam bumbu, aduk hingga rata.
4) Panaskan minyak, lalu masukkan bumbu.
5) Aduk rata, tambahkan gula pasir, lada bubuk, kecap manis, tes rasa.
6) Siap disajikan

Tips:
- Pastikan semua bahan sudah disiapkan sebelum mulai memasak
- Ikuti langkah-langkah dengan teliti untuk hasil terbaik
- Sesuaikan tingkat kepedasan dan rasa sesuai selera
```