Intel Unnati program 2024 :

Problem statement :

“Running GenAI on Intel AI Laptops and Simple LLM Inference on CPU, and Fine-Tuning of LLM Models using Intel® OpenVINO™”

Making Simple LLM model (ChatBot) & Inference with OpenVino on CPU

In [ ]:
```
# Install necessary libraries
# You can run this cell to install required libraries if not already installed
!pip install transformers datasets openvino openvino-dev[tensorflow2,onnx] onnx
```

In [ ]:
```
# Import necessary libraries
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer, TrainingArgu
from datasets import Dataset, load_dataset
from openvino.runtime import Core
import onnx
import os
```

In [ ]:
```
# Load the model and tokenizer
model_name = "TinyLlama/TinyLlama-1.1B-intermediate-step-1431k-3T"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)
```

In [ ]:
```
# Export the model to ONNX
def export_to_onnx(model, tokenizer, onnx_model_path):
    dummy_input = tokenizer("This is a test input", return_tensors="pt").input_ids
    torch.onnx.export(
        model,
        dummy_input,
        onnx_model_path,
        input_names=["input_ids"],
        output_names=["output"],
        dynamic_axes={"input_ids": {0: "batch_size"}, "output": {0: "batch_size"}},
        opset_version=14
    )

onnx_model_path = "model.onnx"
export_to_onnx(model, tokenizer, onnx_model_path)
```

In [ ]:
```
# Verify that the ONNX model was created successfully
assert os.path.exists(onnx_model_path), "ONNX model export failed."
```

In [ ]:
```
# Convert the ONNX model to OpenVINO IR format
mo_command = f"mo --input_model {onnx_model_path} --output_dir . --input_shape [1,1
os.system(mo_command)
```

In [ ]:
```
# Verify that the IR model files were created successfully
assert os.path.exists("model.xml") and os.path.exists("model.bin"), "Model Optimize
```

In [ ]:
```
# Load the OpenVINO model
ie = Core()
compiled_model = ie.compile_model("model.xml", "CPU")
input_layer = compiled_model.input(0).get_any_name()
output_layer = compiled_model.output(0).get_any_name()
```

In [ ]:
```
# Define a function to generate responses with OpenVINO
def generate_response_openvino(prompt, max_length=50):
    inputs = tokenizer(prompt, return_tensors="pt")
    input_ids = inputs.input_ids.numpy()
    output = compiled_model([input_ids])
    output_ids = torch.tensor(output[output_layer]).argmax(dim=-1)
    response = tokenizer.decode(output_ids[0], skip_special_tokens=True)
    return response
```

In [ ]:
```
# Interactive chatbot loop
def chat():
    print("Chatbot is ready! Type 'exit' to end the conversation.")
    while True:
        user_input = input("You: ")
        if user_input.lower() == 'exit':
            print("Goodbye!")
            break
        response = generate_response_openvino(user_input)
        print(f"Bot: {response}")

# To start the chat, simply run the `chat()` function
chat()
```

Fine-tune model with specific dataset (to make custom chatbot) and model work on our data

In [ ]:
```
# Import necessary libraries(optional if already mentioned in above or pip)
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer, TrainingArgu
from datasets import Dataset, load_dataset

# Load your fine-tuning dataset from data.txt
dataset = Dataset.from_file('data.txt')
```

In [ ]:
```
model_name = "TinyLlama/TinyLlama-1.1B-intermediate-step-1431k-3T"
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Tokenize the dataset
def tokenize_function(examples):
    return tokenizer(examples['text'], padding='max_length', truncation=True)

tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

In [ ]:
```
# Define Training Arguments
training_args = TrainingArguments(
    per_device_train_batch_size=4,
    num_train_epochs=3,
    logging_dir='./logs',
    logging_steps=100,
    save_steps=1000,
    output_dir='./results',
)
```

In [ ]:
```
# Define Trainer and Train the Model
model = AutoModelForCausalLM.from_pretrained(model_name)
```

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets['train'],
    eval_dataset=tokenized_datasets['validation'],
)

trainer.train()
```

ChatBot Run after fine tuning

Optional : Want to creat application for model using Flask

Successfully Completed