INTEL UNNATI INDUSTRIAL TRAINING 2024

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

PS04: Introduction to GenAI and Simple LLM Inference on CPU and fine tuning of LLM Model to create a Custom Chatbot

Github Repository Link:
https://github.com/vlen4114/Intel-Unnati-Industrial-Training-2024

Institution Name	SRM Institute of Science and Technology, Chennai		
Team Name	ByteTribe	**Faculty Mentor**	Dr. Prithi Samuel
Team Members	Vasan Lennin (Leader)	**Industry Mentor**	Ms. Vasudha Kumari
	Gudimella Saketa Sri Ramacharyulu	**External Mentor**	Mr.Abhishek Nandy

Problem Statement

Running GenAI on Intel AI Laptops and Simple LLM Inference on CPU and fine-tuning of LLM Models using Intel® OpenVINO™

Our Use Case:

Despite advancements in medical technology, individuals often lack accessible, reliable, and timely information regarding potential health issues and associated precautions. Many people delay seeking medical advice due to the inability to recognize symptoms early or understand their significance. This can lead to the worsening of medical conditions that could have been managed or mitigated with early intervention and proper precautionary measures.

Solution

Symptom-Based Input: Users input their symptoms into the chatbot, which asks targeted questions to gather detailed information.

Custom Dataset: The chatbot leverages a custom-created dataset containing questions and answers related to various diseases, ensuring accurate and comprehensive symptom-disease mapping.

Machine Learning Model: Our LLM is powered by Llama-2-7b. We trained based on the dataset, interprets the user inputs, predicting potential diseases based on the symptoms provided.

Accurate Predictions: The model continuously updates and improves with new data, ensuring precise and reliable disease predictions and medical advice.

Features

- Interactive chatbot interface for user interaction.
- Symptom-based disease prediction using a custom dataset.
- Integration of a machine learning model for accurate predictions.
- Real-time response and recommendations based on user input.
- Scalability and flexibility for future updates and enhancements.

Functional Process

Setup:

- **Install Required Packages:** This involves installing necessary Python packages such as accelerate, peft, bitsandbytes, transformers, and trl. These packages support model training, optimization, quantization, and transformer-based language models.

Configuration:

- **Define Variables:** Set up key variables such as the model name (NousResearch/Llama-2-7b-chat-hf), dataset name (vl8222/Disease_Symptom_Prediction_v5-1k), and the name for the fine-tuned model (Llama-2-7b-chat-finetune-for-disease-prediction). These variables are used throughout the project to specify the model and dataset.

Functional Process

Hugging Face Authorization:

- **Login to Hugging Face Hub:** This step involves logging into the Hugging Face Hub using the command !huggingface-cli login. This authentication is necessary to upload the fine-tuned model to the Hub.

Model Selection:

- **Load the Model and Tokenizer:** Use the AutoModelForCausalLM and AutoTokenizer classes from the transformers library to load the pre-trained Llama-2 model and its tokenizer from the Hugging Face Hub. This sets up the model for further fine-tuning.

Functional Process

Model Size:

- **Load Dataset and Prepare for Training:** Load the dataset specified for fine-tuning using the load_dataset function. This dataset contains samples for disease symptom prediction and is prepared for the training process.

OpenVINO Initialization:

- **Fine-Tune the Model:** Use the SFTTrainer class to fine-tune the model with the loaded dataset. Set various training parameters like the number of epochs, batch size, learning rate, etc. The model is fine-tuned to better predict diseases based on symptoms provided in the dataset.

Functional Process

Model Precision:

- **Configure BitsAndBytes:** Set up the model for 4-bit quantization using BitsAndBytesConfig. This configuration reduces memory usage and increases efficiency without significantly affecting model performance. It allows the model to run more efficiently on GPUs.

Model Conversion:

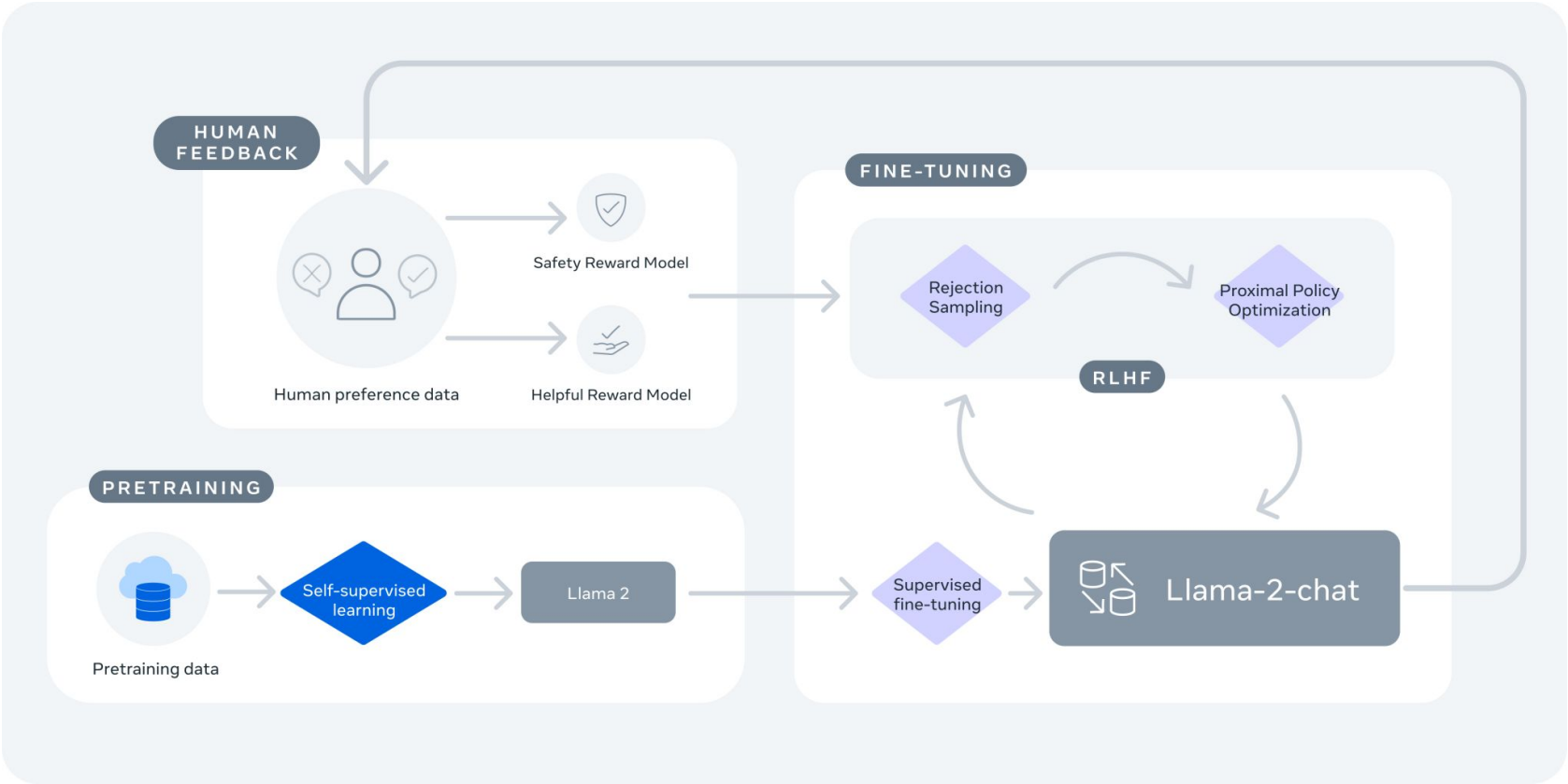
- **Apply Low-Rank Adaptation (LoRA):** Utilize LoraConfig to apply Low-Rank Adaptation, a technique for parameter-efficient fine-tuning. LoRA introduces additional low-rank matrices to the model, which helps in fine-tuning with fewer trainable parameters while maintaining performance.

Functional Process

Model Inference:

- **Generate Text Based on Input Symptoms:** Create a text generation pipeline using the fine-tuned model. Given an input prompt describing symptoms, the model generates text that predicts possible diseases and suggests next steps. For example, "my Symptoms are fatigue cough high fever breathlessness mucoid sputum" might generate a response suggesting possible respiratory infections.

Architecture Diagram



Tech Stack

1) **Core Components:**
   - **Large Language Model (LLM):**
     - **Llama 2 (Fine-tuned):** The foundation, specifically fine-tuned on relevant disease prediction data to enhance its domain-specific knowledge.
   - **Framework:**
     - **Intel Extension for Transformers:** Crucial for optimizing Llama 2 on Intel hardware, enabling faster inference and potentially reduced resource consumption.
     - **Transformers (Hugging Face):** Essential for working with Llama 2 and other transformer-based models.

2) **Programming Language:**
   - **Python:** The primary language for running our LLM, fine tuning, and integrating with most of the libraries and tools.

3) **Core Modules:**
   - **os:** For interacting with the operating system (file/directory operations).
   - **torch (PyTorch):** The fundamental machine learning library for tensor operations and neural network building.
   - **datasets (Hugging Face Datasets):** Facilitates loading and managing datasets.

Tech Stack

3) Core Modules:

- **transformers (Hugging Face Transformers):** The backbone library for working with transformer-based models like Llama 2.
  - AutoModelForCausalLM: Loads pre-trained causal language models.
  - AutoTokenizer: Loads corresponding tokenizers for text preprocessing.
  - BitsAndBytesConfig: Configures quantization (optional) to reduce memory usage.
  - HfArgumentParser: Parses command-line arguments.
  - TrainingArguments: Defines parameters for fine-tuning.
  - pipeline: Simplifies model inference.
  - logging: For logging information and debugging.
- **peft (Parameter-Efficient Fine-Tuning):** Enables efficient fine-tuning.
  - LoraConfig: Configures LoRA for parameter-efficient fine-tuning.
  - PeftModel: Wraps the original model for applying PEFT techniques.
- **trl (Transformer Reinforcement Learning):** Provides tools for reinforcement learning, though used here for supervised fine-tuning (SFT).
  - SFTTrainer: Implements the supervised fine-tuning process.
- **intel_extension_for_transformers (IExT):**
  - Used for optimizations like quantization, pruning, and neural architecture search, specifically targeting Intel hardware for faster and more efficient inference.

Conclusion

In this project, we successfully fine-tuned the Llama-2 model for disease symptom prediction. By leveraging advanced techniques such as Low-Rank Adaptation (LoRA) and 4-bit quantization through the BitsAndBytes library, we optimized the model's performance and efficiency. The project involved loading a pre-trained Llama-2 model, configuring it with necessary quantization settings, and fine-tuning it on a specialized dataset for disease symptom prediction. After fine-tuning, we validated the model's capability to generate relevant disease predictions based on input symptoms. The final model, along with its tokenizer, was uploaded to the Hugging Face Hub, making it easily accessible for deployment and further usage. This project demonstrates the potential of transformer-based language models in the healthcare domain, providing a foundation for developing advanced predictive tools for disease diagnosis and management.

Team Members and Contributions

Saketa Sri Ramacharyulu Gudimella

- Installed required packages.
- Configured model and quantization settings.
- Loaded and prepared the Llama-2 model.
- Created and validated the text generation pipeline.

Vasan Lennin

- Loaded and preprocessed the dataset.
- Set up and conducted model fine-tuning with Llama-2 model.
- Uploaded the model to the Hugging Face Hub.
- Documented the project and prepared the report.