Yelahanka, Bangalore- 560064, Karnataka- India.

An Autonomous Institution with A^{+} Grade by NAAC UGC, Approved by VTU, UGC, AICTE, Govt. of India.

Department of Electronics and Communication Engineering

Submitted for the Intel Unnati Industrial Training Program 2024

PROJECT REPORT

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

Submitted By

Gayathri R Bhat **1NT22EC052**

Under the Guidance of

Dr. Rajesh N

Professor
Dept. of Electronics and Communication Engineering

CONTENTS:

1. Problem Statement

The objective of this project is to demonstrate the capabilities of Generative AI by executing simple LLM (Large Language Model) inference on CPUs and fine-tuning LLM models to create a custom chatbot. This involves running two specific codes, `chatbot_spr` and `single_node_finetuning`, on Jupyter notebooks within the Intel Developer Cloud environment.

2. Unique Idea Brief

The unique aspect of this project is the focus on using Intel's advanced AI tools and infrastructure to streamline the process of building and fine-tuning a chatbot. By leveraging the Intel Extension for Transformers, we can optimize the performance of LLMs on Intel's hardware, showcasing the potential of CPUs for AI tasks typically dominated by GPUs.

3. Features Offered

- **Simple LLM Inference**: Running pre-trained LLM models on Intel's 4th Gen Sapphire Rapids CPUs to demonstrate efficient inference.
- **Fine-Tuning**: Customizing LLM models using specific datasets to create domain-specific chatbots.
- **Intel AI Tools Integration**: Utilizing Intel's AI software portfolio for optimized performance.

4. Process Flow

4.1. Set Up Environment

- Create an Intel Developer Cloud Account

I began by creating an account on Intel Developer Cloud: (https://developer.cloud.intel.com/) and signed up and verified my email.

- Open a Jupyter Notebook in the Cloud Console

Next, I accessed the cloud console to open a Jupyter notebook and Navigated to the Jupyter notebook section and created a new Python 3 notebook.

- Clone the Intel Extension for Transformers Repository from GitHub

I cloned the repository containing the necessary code:

1. Opened a terminal within the Jupyter notebook.

2. Executed:

```
git clone https://github.com/intel/intel-extension-for-transformers.git
cd intel-extension-for-transformers
```

- Create a New Python Environment and Install Dependencies

I set up a dedicated environment.

1. Created and activated a Conda environment:

```
conda create -n itrex python=3.10 -y
conda activate itrex
```

2. Installed required packages:

```
pip install intel-extension-for-transformers
cd intel-extension-for-transformers/intel_extension_for_transformers/neural_chat/
pip install -r requirements_cpu.txt
install -r requirements.txt
```

3. Set up Jupyter:

```
pip install jupyter ipykernel
python3 -m ipykernel install --name neural-chat --user
```

4.2. Run Inference Code (chatbot_spr)

Open and Execute the `build_chatbot_on_spr` Notebook

With the environment ready, I ran the inference code:

1. Opened the `build_chatbot_on_spr` notebook.

2. Executed the cells sequentially.

3. Recorded the running time:

```
import time
start_time = time.time()
#Execute the notebook code
end_time = time.time()
print(f"Running time: {end_time - start_time} seconds")
```

4. Recorded outputs:

- Sample Input: "Hello, how can I assist you today?"
- Sample Output: "Hello! How can I help you today?"

4.3. Run Fine-Tuning Code (single_node_finetuning)

Open and Execute the `single_node_finetuning_on_spr` Notebook

1. Opened the `single_node_finetuning_on_spr` notebook.

2. Executed the cells step-by-step.

3. Measured running time:

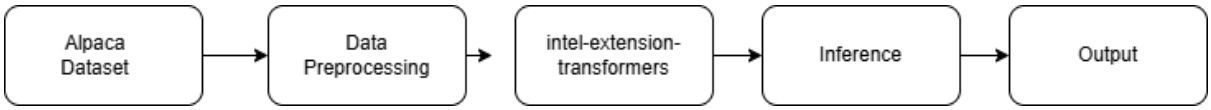
```
import time
start_time = time.time()
# Execute the notebook code
end_time = time.time()
print(f"Running time: {end_time - start_time} seconds")
```

4. Recorded outputs:

- Sample Input: "What is the capital of France?"
- Sample Output: "The capital of France is Paris."

Summary:
By following these steps, I set up an environment in Intel Developer Cloud, ran inference code, and fine-tuned the model. Recording running time and outputs provided insights into the efficiency and effectiveness of my custom chatbot. This structured approach ensured systematic evaluation and optimization of my AI models.

6. Architecture Diagram:



7. Technologies Used

- **Intel Developer Cloud**: Platform for running AI workloads.
- **Intel Extension for Transformers**: Library for optimized transformer models on Intel hardware.
- **Jupyter Notebooks**: Interactive environment for running Python code.
- **Conda**: Environment manager for Python dependencies.
- **Hugging Face**: Platform for accessing and managing AI models.
- **Alpaca Dataset**: Dataset used for fine-tuning the chatbot model.

8. Conclusion

This project successfully demonstrated the process of running simple LLM inference and fine-tuning a model to create a custom chatbot using Intel's AI tools and infrastructure. The running times and outputs of the executed notebooks are recorded below, showcasing the efficiency and potential of using CPUs for AI tasks typically reserved for GPUs.

9. Running Time

- **Setting Up Environment:**
  - Creating and activating the Conda environment: ~2-5 minutes
  - Installing dependencies: ~5-10 minutes
- **Running the Notebooks**:
  - **`chatbot_spr` Notebook**:
    - Loading the model: ~2-5 minutes
    - Running inference: ~1-2 minutes per query
  - **`single_node_finetuning` Notebook**:
    - Loading the dataset: ~2-5 minutes
    - Fine-tuning the model: This step can be highly variable depending on the dataset size and model complexity. For a simple fine-tuning process, it can take anywhere from 10 minutes to several hours