Introduction to GenAI and Simple LLM Inference on CPU and finetuning of LLM Model to create a Custom Chatbot

Intel® Unnati Industrial Training Project,2024

By
19Bits

Joshua Sunny Ninan (B23CS1140)
Rohith N S (B23CS2156)
Avin Joy (B23CS2123)
Edwin K Mathew (B23CS1128)

Mentored by

Dr. Tessy Mathew

MAR BASELIOS COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous)

MAR IVANIOS VIDYA NAGAR, NALANCHIRA, THIRUVANANTHAPURAM, 695 015

MAY, 2024

ABSTRACT

Large Language Models, or LLMs, are a type of Generative Artificial Intelligence (GenAI), and this research investigates their potential for commercial use. It looks into whether LLM inference can be performed on CPUs that are easily accessible and shows how to optimize an LLM to build a personalized chatbot.

An overview of GenAI principles and LLM capabilities is provided in the report. After that, it looks at methods for effective LLM inference on CPU architectures, emphasizing the benefits of CPU-based solutions in business contexts. Lastly, the project shows how to refine a pre-trained LLM using a Stanford alpaca dataset to create a unique chatbot. The possible advantages and factors to take into account while implementing GenAI and customized chatbots in industrial settings are covered in the report's conclusion.

INTRODUCTION:

The frontiers of Artificial Intelligence (AI) are rapidly expanding, with Generative AI (GenAI) and Large Language Models (LLMs) leading the charge. GenAI pushes the boundaries of content creation, crafting entirely new forms of text, music, and even code. Imagine a system analyzing vast amounts of music to compose a fresh song inspired by the style. That's the power of GenAI! It devours massive datasets, learning the underlying patterns and relationships within the data. This allows GenAI models to transcend mere imitation, generating entirely novel outputs fueled by their training.

LLMs, a specialized form of GenAI, excel at manipulating and generating text. Think of them as incredibly powerful language experts. Trained on colossal amounts of text data, they perform impressive feats like generating human-quality text formats (news articles, code!), translating languages with remarkable accuracy, and summarizing lengthy documents. Imagine providing an LLM a simple prompt and receiving a complete news story! LLMs can even act as a powerful knowledge source, offering informative answers to your questions.

This report dives into the core principles of GenAI and LLM functionalities. This exploration will establish a foundation in these concepts, delve into the growing importance of CPU-based LLM inference methods, and showcase a practical application: fine-tuning pre-trained LLMs for specific tasks. This structured approach equips you with a comprehensive understanding of these groundbreaking AI tools.

LITERATURE REVIEW

Generative Artificial Intelligence (GenAI) and Large Language Models (LLMs):

Generative Artificial Intelligence (GenAI) has emerged as a potent tool in the field of Artificial Intelligence (AI), which has seen a notable movement towards generative models. Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) are two examples of approaches that fall under the umbrella of GenAI. These techniques have the ability to produce unique data points and learn underlying data distributions. Large Language Models (LLMs) have attracted a lot of interest in GenAI because of their capacity to process and produce text that is similar to that of a human.

To accomplish tasks like text production, translation, question answering, and code authoring, LLMs are essentially neural networks that have been trained on large volumes of text data. Research has demonstrated that LLMs can produce a wide range of realistic and coherent text formats, including emails, screenplays, musical compositions, and poems. Because of its adaptability, LLMs are being used in a variety of industries. Examples of these include automated customer support chatbots and content creation assistants for creative writing.

Challenges and Considerations for Industrial Applications:

Despite the remarkable capabilities of LLMs, deploying them in industrial settings presents two main challenges: computational cost and customization.

Computational Cost: Because of the intricate computations required, traditional LLM inference—running the model to produce outputs for fresh data points—heavily depends on strong Graphics Processing Units (GPUs). Although GPUs perform better, their high cost and power consumption prevent them from being used in industrial settings with limited resources.

Customization: Although adaptable, pre-trained LLMs might not be the best fit for certain industrial requirements. Large datasets that may not contain the domain-specific knowledge needed for commercial applications are used to train these models. This lack of customization may make LLMs less efficient at handling problems and duties unique to a certain sector.

CPU-based LLM Inference:

The limits of GPU-based inference have prompted studies on executing LLMs on Central Processing Units (CPUs) that are widely accessible. GPUs are more suitable for deep learning workloads than traditional CPU architectures. To optimize LLM inference for CPUs, however, recent research has looked into methods like quantization and knowledge distillation. While knowledge distillation includes moving knowledge from a pre-trained teacher model (often a GPU-based LLM) to a smaller student model (designed for CPU), quantization entails decreasing the precision of the model weights. Thanks to these methods, deploying LLMs in industrial settings is feasible as long as CPUs remain widely available and affordable.

Fine-tuning LLMs for Chatbot Development:

Customization gaps can be filled in part by optimizing previously trained LLMs. A pre-trained model is fine-tuned by applying it to a new, smaller dataset that is unique to the intended job. In the context of industrial applications, this can entail optimizing an LLM using a dataset that includes conversation data unique to a given industry in order to build a personalized chatbot. After that, the chatbot might be employed for customer service, question-answering, or even helping with process control in a manufacturing environment. Promising outcomes from research on optimizing LLMs for chatbot creation have opened the door for more specialized and efficient chatbots across a range of sectors.

PROBLEM STATEMENT

Problem Statement (PS-04): Introduction to GenAI and Simple LLM Inference on CPU and finetuning of LLM Model to create a Custom Chatbot

METHODOLOGY

Methodology for Fine-Tuning TinyLLama on Intel Xeon SPR

1. Environment Setup

Objective: Set up a Python environment with the necessary dependencies.

Steps:

1. **Create and activate a Conda environment:**

```
conda create -n itrex-1 python=3.10 -y
conda activate itrex-1
```

This creates a new Conda environment named `itrex-1` with Python 3.10 and activates it.

2. **Install required Python packages:**

```
pip install intel-extension-for-transformers
```

2. Cloning the Repository

Objective: Clone the GitHub repository containing the fine-tuning scripts.

Steps:

1. **Clone the repository:**

```
git clone https://github.com/eternalflame02/Single-Node-FInetuning-of-Tiny-LLama-using-Intel-Xeon-SPR.git
```

2. **Navigate to the fine-tuning directory:**

```
cd./Single-Node-FInetuning-of-Tiny-LLama-using-Intel-Xeon-SPR/Fine Tuning/
```

3. Installing Additional Dependencies

Objective: Install additional dependencies required for fine-tuning.

Steps:

1. **Install dependencies from the `requirements.txt` file:**

```
pip install -r requirements.txt
```

2. **Install Jupyter and IPython kernel:**

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

4. Setting Up Hugging Face Authentication

Objective: Authenticate with Hugging Face to access and download models.

Steps:

1. **Login to Hugging Face:**

```
huggingface-cli login
```

5. Downloading Data

Objective: Download the dataset required for fine-tuning.

Steps:

1. **Download the Alpaca dataset:**

```
Curl -O https://github.com/tatsu-lab/stanford_alpaca/raw/main/alpaca_data.json
```

6. Fine-Tuning the Model

Objective: Fine-tune the TinyLLama model using the downloaded dataset.

Steps:

1. **Import necessary modules:**

```
python

from transformers import TrainingArguments
from intel_extension_for_transformers.neural_chat.config import (
    ModelArguments,
    DataArguments,
    FinetuningArguments,
    TextGenerationFinetuningConfig,
)
from intel_extension_for_transformers.neural_chat.chatbot import finetune_model
import os
```

2. **Define model, data, and training arguments:**

```
python

# Define model arguments
model_args                                                    =
ModelArguments(model_name_or_path="TinyLlama/TinyLlama-
1.1B-Chat-v1.0")

# Define data arguments
data_args   =   DataArguments(train_file="alpaca_data.json",
validation_split_percentage=1)

# Define training arguments
training_args = TrainingArguments(
    output_dir='tinyllama',
    overwrite_output_dir=True,
    do_train=True,
    do_eval=True,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    gradient_accumulation_steps=2,
    learning_rate=4e-5,  # Adjust learning rate
    weight_decay=0.01,  # Add weight decay
    num_train_epochs=1,    #  Increase  epochs  for  better
training
    save_steps=500,
    save_total_limit=2,
    logging_steps=100,
    evaluation_strategy="steps",
    load_best_model_at_end=True,
    seed=42,  # Set random seed
    bf16=True,
    resume_from_checkpoint=True    #  Load  from  the  latest
checkpoint if finetuning fails after creating a checkpoint
)

# Define finetuning arguments
finetune_args = FinetuningArguments()

# Create finetuning configuration
finetune_cfg = TextGenerationFinetuningConfig(
    model_args=model_args,
    data_args=data_args,
    training_args=training_args,
    finetune_args=finetune_args,
)
```

3. **Start the fine-tuning process:**

```
python

finetune_model(finetune_cfg)
```

7. Building the Chatbot

Objective: Build a chatbot using the fine-tuned model.

Steps:

1. **Import necessary modules for building the chatbot:**

```
python

from intel_extension_for_transformers.neural_chat import
build_chatbot
from intel_extension_for_transformers.neural_chat import
PipelineConfig
from intel_extension_for_transformers.neural_chat.config
import LoadingModelConfig
```

2. **Create a pipeline configuration:**

```
python

config = PipelineConfig(
    model_name_or_path="./tinyllama",
    loading_config=LoadingModelConfig(
        peft_path="./tinyllama"   # Path to the PEFT model
(fine-tuned model)
    )
)
```

3. **Build the chatbot instance:**

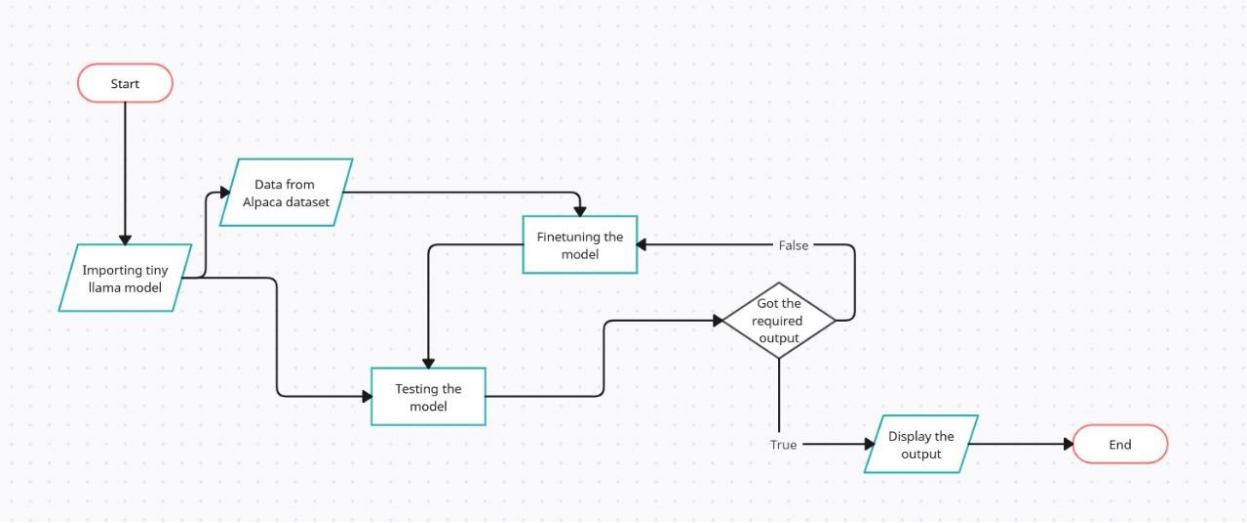
```
python

chatbot = build_chatbot(config)
```

4. **Generate a response to a query:**

```
python

query = "Tell me about AI."
response = chatbot.predict(query=query)

# Print the generated response
```

```
print(response)
```

DATA FLOW DIAGRAM



RESULTS:

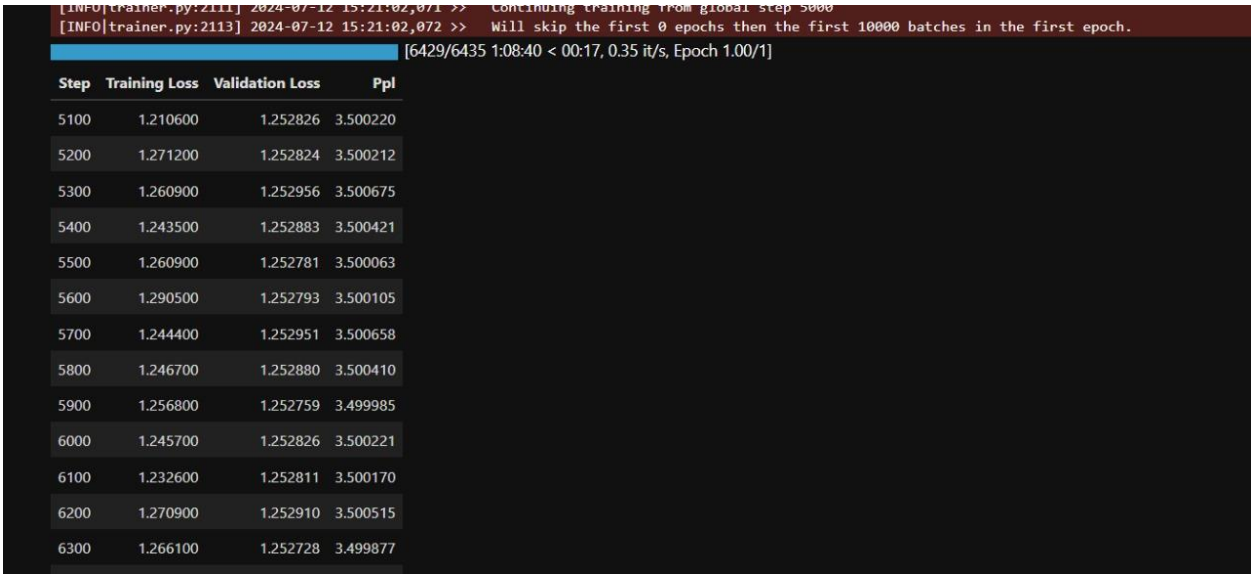
- The fundamental concepts of Transformers, Generative AI, and its applications are understood.
- This project involved performing a simple LLM inference on a CPU and understood the process of fine-tuning LLMs for custom applications.
- Result from the **build_chatbot_on_spr.ipynb** notebook :

```
/home/u9afe94d180edee9c2d70b97a7a50e80/.local/lib/python3.9/site-packages/torchvision/io/image.py:13: UserWa
rning: Failed to load image Python extension: ''If you don't plan on using image functionality from `torchvi
sion.io`, you can ignore this warning. Otherwise, there might be something wrong with your environment. Did
you have `libjpeg` or `libpng` installed before building `torchvision` from source?
  warn(
/home/u9afe94d180edee9c2d70b97a7a50e80/.local/lib/python3.9/site-packages/transformers/deepspeed.py:23: Futu
reWarning: transformers.deepspeed module is deprecated and will be removed in a future version. Please impor
t deepspeed modules directly from transformers.integrations
  warnings.warn(
/home/u9afe94d180edee9c2d70b97a7a50e80/.local/lib/python3.9/site-packages/huggingface_hub/file_download.py:1
132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. Downloads always r
esume when possible. If you want to force a new download, use `force_download=True`.
  warnings.warn(
Loading model Intel/neural-chat-7b-v3-1
Loading checkpoint shards: 100% 2/2 [00:08<00:00, 4.13s/it]
A CPU, or Central Processing Unit, is like the brain of a computer. It's responsible for executing instructi
ons and processing data received from other components such as memory and input devices. CPUs are designed t
o handle complex tasks quickly and efficiently, making them essential for the smooth functioning of modern c
omputers and electronic gadgets. интелектуальный процессор является сердцем компьютера, отвечая за выполнени
е команд и обработку данных, полученных от других компонентов, таких как память и входные устройства. Интелл
ектуальные процессоры предназначены для быстрого и эффективного решения сложных задач, что делает их необход
имыми для работы современных компьютеров и электронных устройств. интелектуальный процессор є серцем комп'ют
ера, що відповідає за виконання команд і обробку даних, отриманих від інших компонентів, наприклад, па

A Transformer is a type of neural network architecture designed for natural language processing tasks. It wa
s introduced in 2017 by researchers at Google AI. The model has shown remarkable performance in various NLP
```

```
tasks such as machine translation, text summarization, and question answering. Its ability to learn long-ter
m dependencies between words makes it particularly effective in understanding complex contexts within senten
ces or documents.

In simpler terms, imagine a Transformer like a superhero with the power to understand different languages an
d texts, making it easier for machines to communicate with humans and translate from one language to anothe
r. індествильно індествильно індествильно індесівяльно індесівяльно індесівяльно індесівяльно індесівяльно і
ндесівяльно індесівяльно індесівяльно індесівяльно індесівяльно індесівяльно індесівяльно індесівяльно індес
івяльно індесівяльно індесівяльно індесівяльно індесівяль

A GPU, or Graphics Processing Unit, is a specialized electronic circuit designed for accelerating the render
ing of graphics and images on various devices like computers, smartphones, and gaming consoles. It works in
parallel with the Central Processing Unit (CPU) to handle complex visual tasks efficiently, making it essent
ial for high-quality video games, virtual reality experiences, and advanced computer simulations. In summar
y, a GPU is like a superhero for your screen, bringing life to stunning visuals. інде інде інде інде інде ін
де інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде
інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інд
е інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде інде і
нде інде інде інде ін

An AI Chatbot is an artificially intelligent system designed to simulate human conversation through text or
voice interactions. It can understand natural language, respond with relevant information, engage in convers
ations, and even exhibit emotional responses. These virtual assistants aim to provide assistance, entertainm
ent, and companionship to users while continuously learning and adapting to their needs. инде инде инде инде
инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инд
е инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде и
нде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде
инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде инде
```

```
A processor, also known as a central processing unit (CPU), is the brain of a computer system. It's responsi
ble for executing instructions and managing various tasks within the device. Think of it like the conductor
of an orchestra, coordinating all the different components to work together harmoniously. The processor's sp
eed and efficiency play a crucial role in determining how fast and smoothly your computer can perform tasks.
интелектуальный процессор является сердцем компьютера, отвечая за выполнение команд и управление различными
задачами в системе. Он как дирижёр оркестра, который координирует работу всех компонентов для того, чтобы он
и функционировали вместе эффективно и быстро. Скорость и эффективность процессора определяют сколько и каким
образом ваш компьютер может выполнять задачи. интеллектуальный процессор является сердцем компьютера, отвеча
я за выполнение команд и управление различными за
```

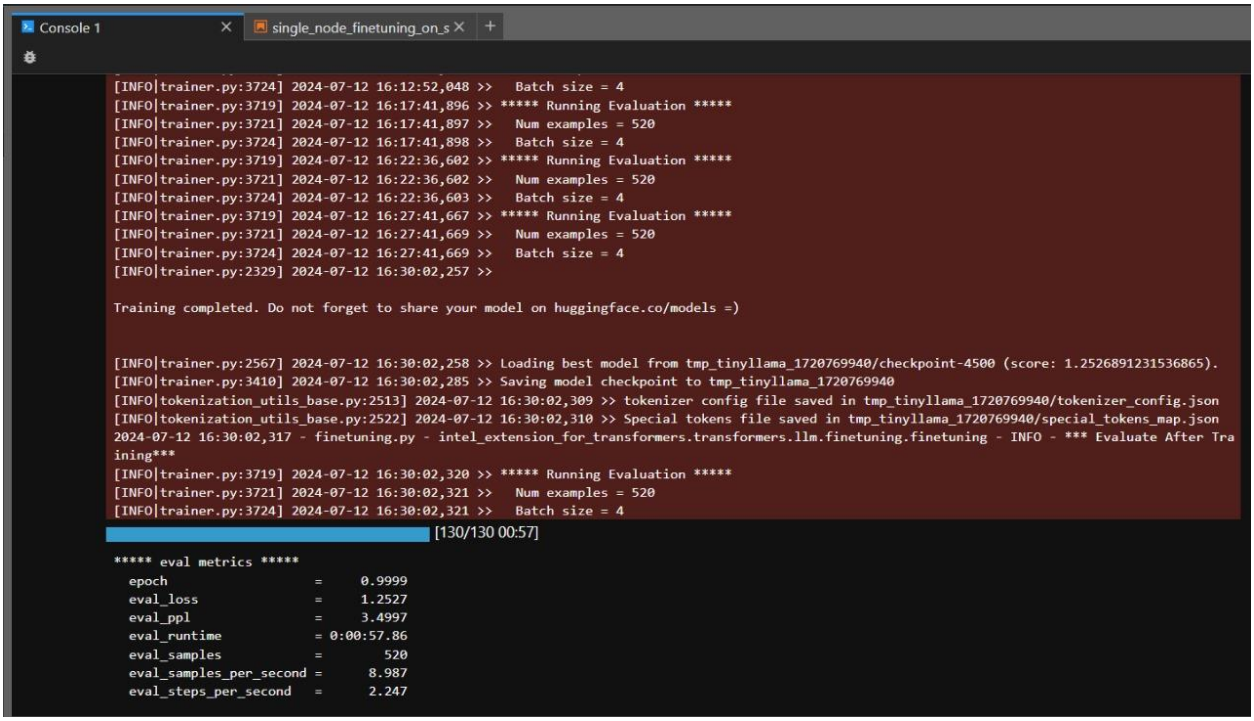
- Result from **single_node_finetuning_on_spr.ipynb** notebook



```
[INFO|trainer.py:2111] 2024-07-12 15:21:02,071 >>   Continuing training from global step 5000
[INFO|trainer.py:2113] 2024-07-12 15:21:02,072 >>   Will skip the first 0 epochs then the first 10000 batches in the first epoch.
```

[6429/6435 1:08:40 < 00:17, 0.35 it/s, Epoch 1.00/1]

Step	Training Loss	Validation Loss	Ppl
5100	1.210600	1.252826	3.500220
5200	1.271200	1.252824	3.500212
5300	1.260900	1.252956	3.500675
5400	1.243500	1.252883	3.500421
5500	1.260900	1.252781	3.500063
5600	1.290500	1.252793	3.500105
5700	1.244400	1.252951	3.500658
5800	1.246700	1.252880	3.500410
5900	1.256800	1.252759	3.499985
6000	1.245700	1.252826	3.500221
6100	1.232600	1.252811	3.500170
6200	1.270900	1.252910	3.500515
6300	1.266100	1.252728	3.499877



```
Console 1   single_node_finetuning_on_s
[INFO|trainer.py:3724] 2024-07-12 16:12:52,048 >>   Batch size = 4
[INFO|trainer.py:3719] 2024-07-12 16:17:41,896 >> ***** Running Evaluation *****
[INFO|trainer.py:3721] 2024-07-12 16:17:41,897 >>   Num examples = 520
[INFO|trainer.py:3724] 2024-07-12 16:17:41,898 >>   Batch size = 4
[INFO|trainer.py:3719] 2024-07-12 16:22:36,602 >> ***** Running Evaluation *****
[INFO|trainer.py:3721] 2024-07-12 16:22:36,602 >>   Num examples = 520
[INFO|trainer.py:3724] 2024-07-12 16:22:36,603 >>   Batch size = 4
[INFO|trainer.py:3719] 2024-07-12 16:27:41,667 >> ***** Running Evaluation *****
[INFO|trainer.py:3721] 2024-07-12 16:27:41,669 >>   Num examples = 520
[INFO|trainer.py:3724] 2024-07-12 16:27:41,669 >>   Batch size = 4
[INFO|trainer.py:2329] 2024-07-12 16:30:02,257 >>

Training completed. Do not forget to share your model on huggingface.co/models =)


[INFO|trainer.py:2567] 2024-07-12 16:30:02,258 >> Loading best model from tmp_tinyllama_1720769940/checkpoint-4500 (score: 1.2526891231536865).
[INFO|trainer.py:3410] 2024-07-12 16:30:02,285 >> Saving model checkpoint to tmp_tinyllama_1720769940
[INFO|tokenization_utils_base.py:2513] 2024-07-12 16:30:02,309 >> tokenizer config file saved in tmp_tinyllama_1720769940/tokenizer_config.json
[INFO|tokenization_utils_base.py:2522] 2024-07-12 16:30:02,310 >> Special tokens file saved in tmp_tinyllama_1720769940/special_tokens_map.json
2024-07-12 16:30:02,317 - finetuning.py - intel_extension_for_transformers.transformers.llm.finetuning.finetuning - INFO - *** Evaluate After Tra
ining***
[INFO|trainer.py:3719] 2024-07-12 16:30:02,320 >> ***** Running Evaluation *****
[INFO|trainer.py:3721] 2024-07-12 16:30:02,321 >>   Num examples = 520
[INFO|trainer.py:3724] 2024-07-12 16:30:02,321 >>   Batch size = 4
[130/130 00:57]

***** eval metrics *****
  epoch                   =     0.9999
  eval_loss               =     1.2527
  eval_ppl                =     3.4997
  eval_runtime            = 0:00:57.86
  eval_samples            =        520
  eval_samples_per_second =      8.987
  eval_steps_per_second   =      2.247
```

CONCLUSION

This project explored the potential of Generative AI (GenAI), specifically Large Language Models (LLMs), for industrial applications. A successful simple LLM inference was achieved on a CPU, showcasing the potential for LLM deployment in resource-constrained environments. Additionally, we gained a practical understanding of the fine-tuning process, culminating in the development of a custom chatbot.

The findings suggest that CPU-based LLM inference, coupled with fine-tuning techniques, can bridge the gap between cutting-edge GenAI technology and practical industrial applications. This approach offers a cost-effective and accessible solution for leveraging the power of LLMs in various industrial settings.

1. REFERENCES

Hugging Face – The AI community building the future.

https://github.com/intel/intel-extension-for-transformers/

https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0

https://huggingface.co/meta-llama/Llama-2-7b-chat-hf