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

**Introduction :**

Fine-tuning a pre-trained Large Language Model (LLM) is a powerful way to create a custom chatbot that can better understand and respond to user input. Models like GPT-4 are trained on a vast array of topics, making them excellent at generating text. However, they can be too general for specialized tasks. Fine-tuning makes these models more specific and useful for particular needs.

### Purpose of Fine-tuning LLM Models for Custom Chatbots

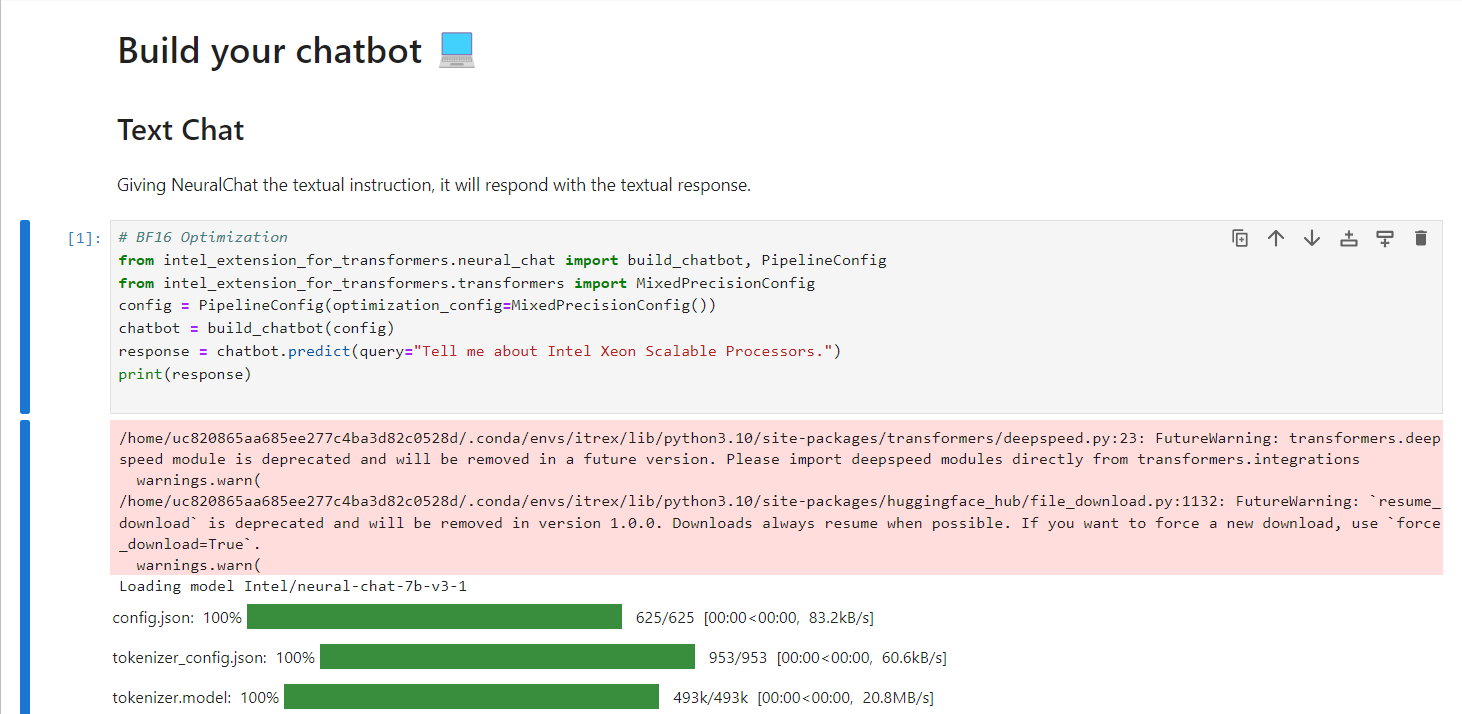
The main purpose of fine-tuning LLM models to create custom chatbots is to enhance their relevance, accuracy, and effectiveness for specific uses.

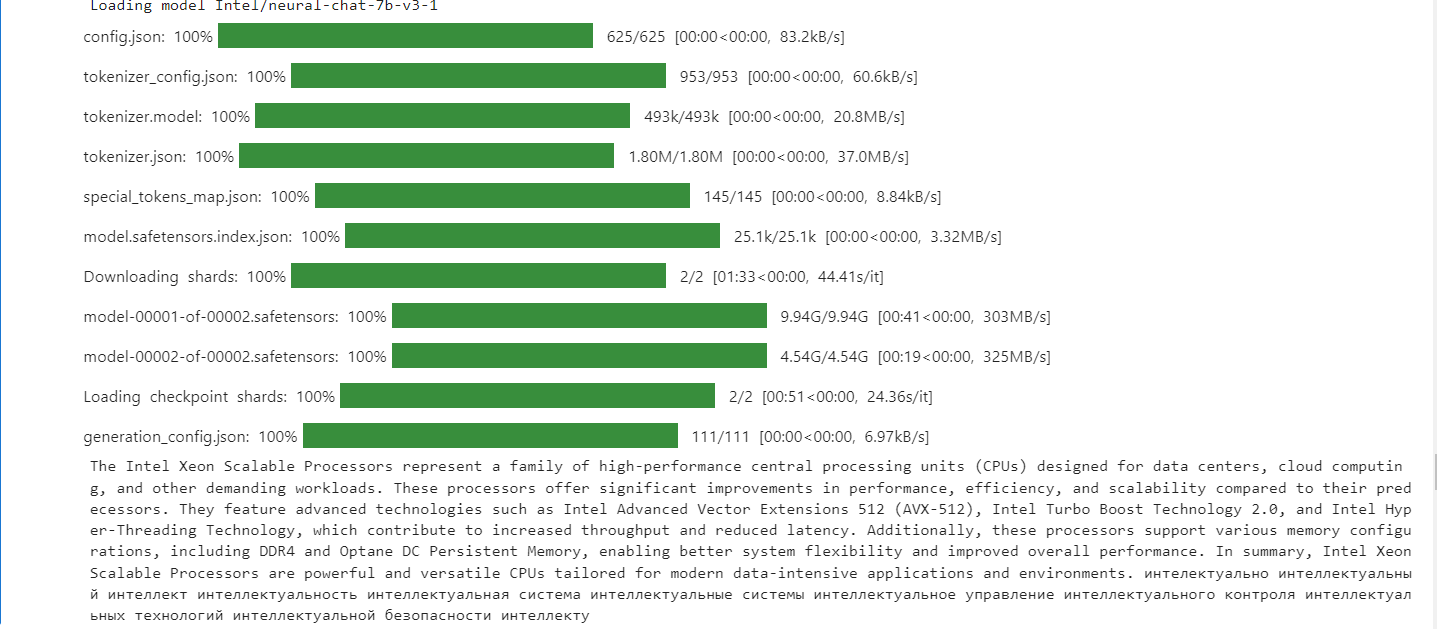
By fine-tuning the model, we can tailor it to a particular domain, task, or tone, improving its ability to handle user queries and provide relevant responses.

**Solutions:**

What are the five texts/questions provided to the model?

1. Tell me about Intel Xeon Scalable Processors





Time taken:

Time taken for each step in the loading process is:

**Loading Configuration and Tokenizer Files**:

* config.json : 00:00<00:00, 83.2KB/s
* tokenizer\_config.json : 00:00<00:00, 60.6KB/s
* tokenizer.model : 00:00<00:00, 20.8MB/s
* tokenizer.json : 00:00<00`:00, 37.0MB/s
* special\_tokens\_map.json : 00:00<00:00, 8.84KB/s
* model.safetensors.index.json : 00:00<00:00, 3.32MB/s

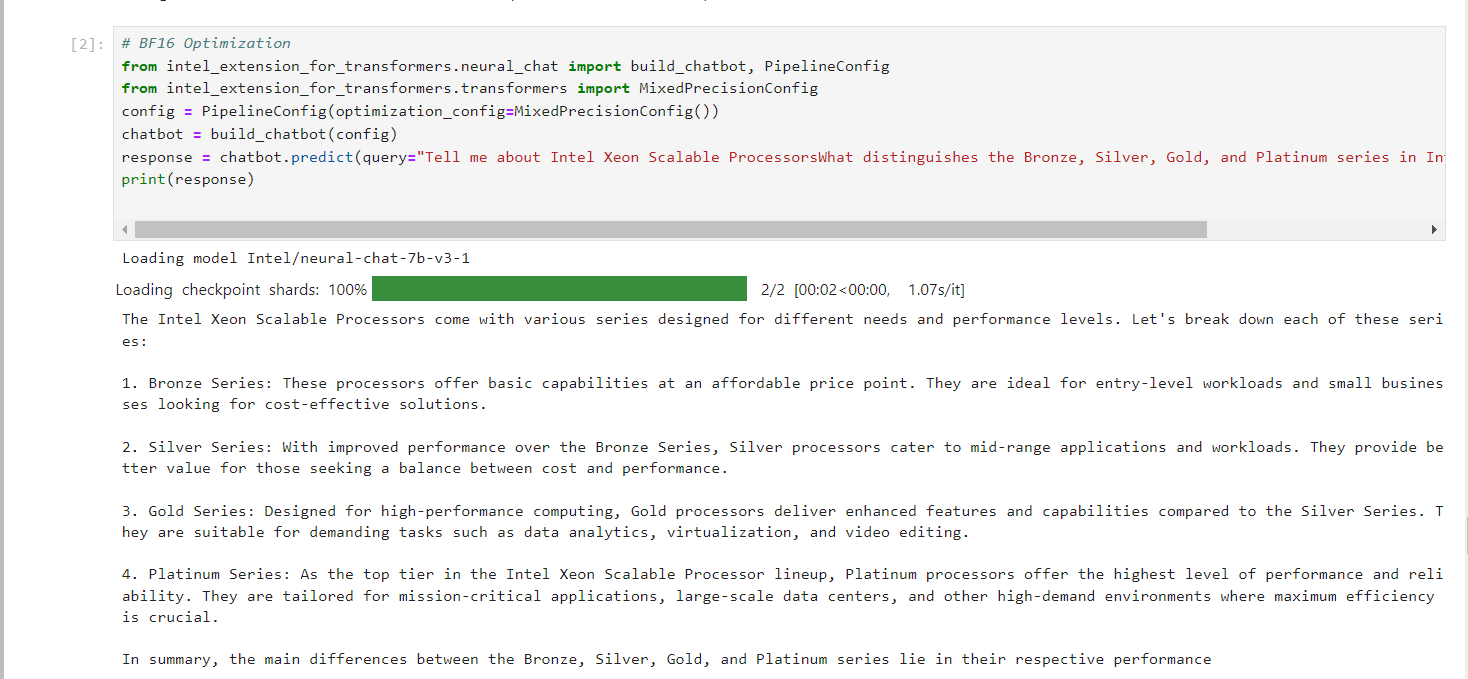
**Downloading and Loading Model Shards**:

* Downloading Shards: 1:33<00:00, 44.41s/it
* model-00001-of-00002.safetensors: 0:41<00:00, 303MB/s
* model-00002-of-00002.safetensors: 0:19<00:00, 325MB/s
* Loading Checkpoint Shards: 0:51<00:00, 24.36s/it

**Loading Generation Configuration**:

* generation\_config.json : 00:00<00:00, 6.97KB/s

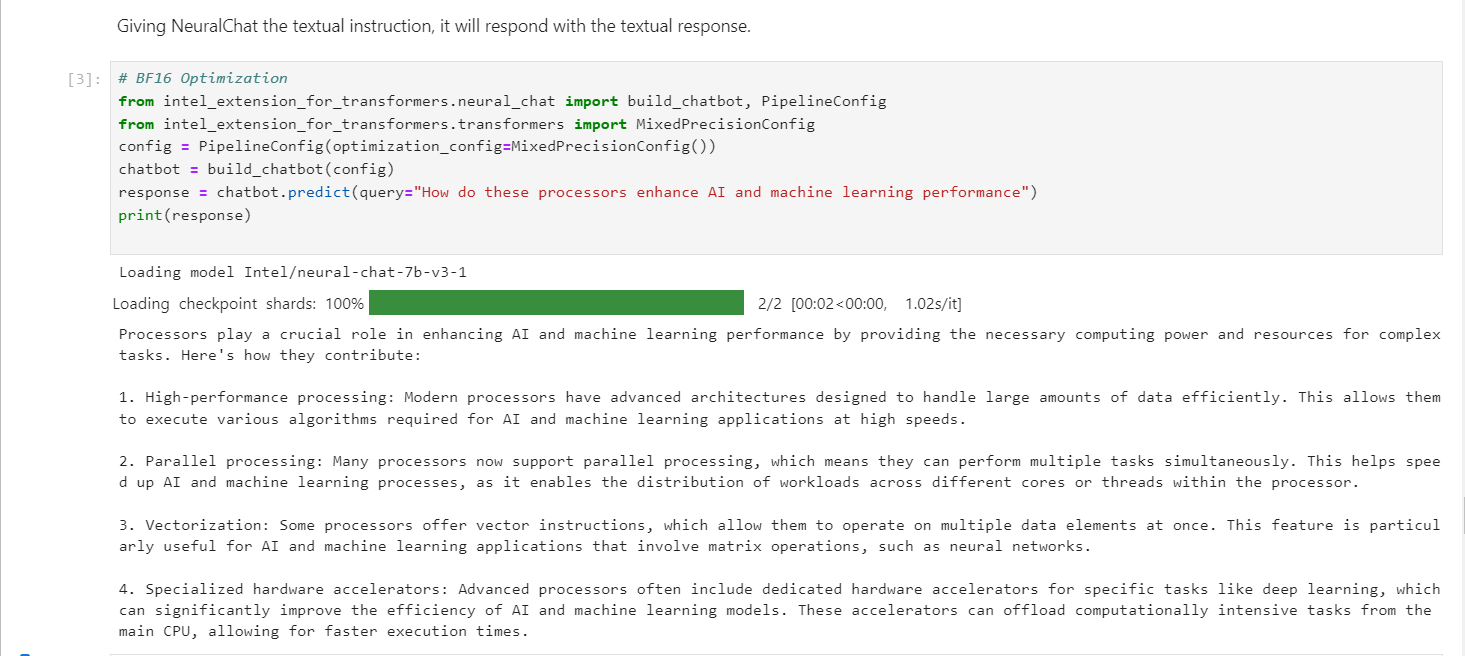
1. What are the key differences between the various series (Bronze, Silver, Gold, and Platinum) in the Intel Xeon Scalable Processor family?



Time taken: 00:02<00:00, 1.07s/it

The time taken for loading the checkpoint shards is approximately 2 seconds (1.07s per iteration for 2 iterations).

1. How do Intel Xeon Scalable Processors enhance AI and machine learning performance?



Time taken: 00:02<00:00, 1.02s/it

The time taken for loading the checkpoint shards is approximately 2 seconds (1.02s per iteration for 2 iterations).

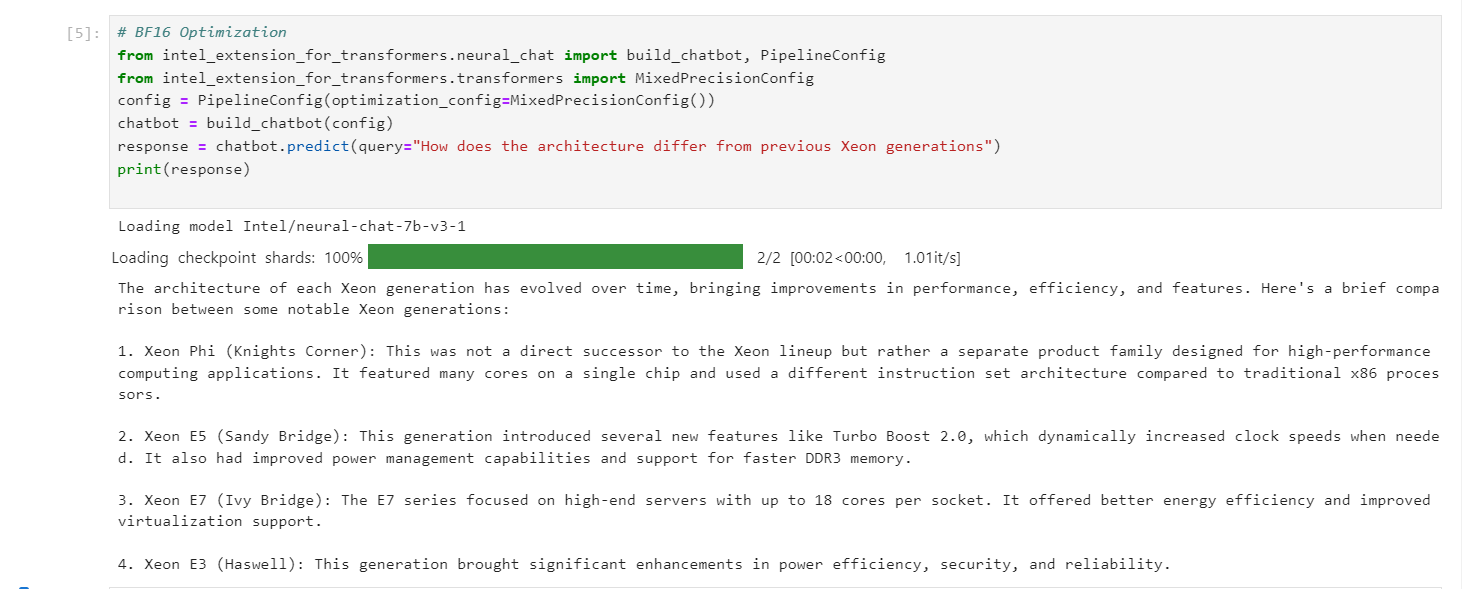
1. What are the security features incorporated in Intel Xeon Scalable Processors?



Time taken: 00:02<00:00, 1.02it/s

The time taken for loading the checkpoint shards is approximately 2 seconds (1.02s per iteration for 2 iterations).

1. How does the architecture of Intel Xeon Scalable Processors differ from previous generations of Xeon processors?



Time taken: 00:02<00:00, 1.01s/it

The time taken for loading the checkpoint shards is approximately 2 seconds (1.01s per iteration for 2 iterations).

**Large Language Models (LLMs)**

Large Language Models (LLMs) are a type of artificial intelligence model designed to understand and generate human language.

LLMs involves using pre-trained models to generate text, while fine-tuning adjusts these models on specific datasets to improve task performance, useful for creating custom chatbots and specialized applications.

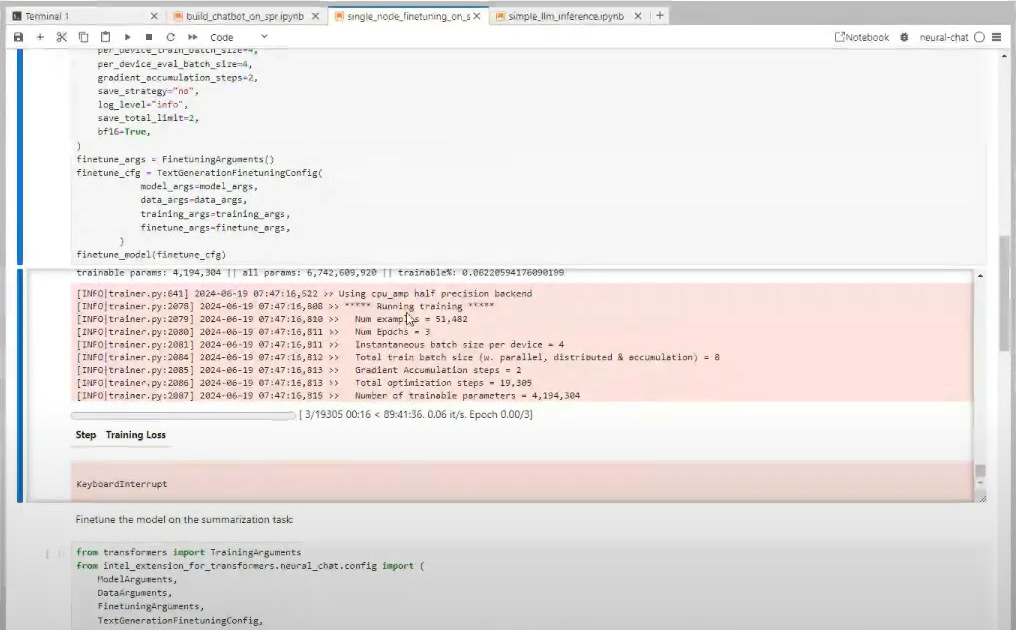
Large language models are trained to solve common language problems, like ...

* Text Classification
* Question-answering
* Document summarization
* Text generation

**Fine-Tuning of LLM Model**

Fine-Tuning Report: Enhancing meta-llama/Llama-2-7b-chat-hf Model

This project aimed to optimize the meta-llama/Llama-2-7b-chat-hf model for summarization tasks using the alpaca\_data.json dataset. The training utilized Intel Extension for Transformers, configuring the model for 3 epochs with a batch size of 4 per device and BF16 precision. The total trainable parameters were 4,194,304 out of 6,742,609,920, with training conducted on a CPU backend.



Training Time: Approximately 89 hours, 41 minutes, and 36 seconds [89:41:36]

Dataset Size: 51,482 examples

**How does Generative AI contribute to our project?**

Generative AI can really boost the project in several ways. It allows for the customization and personalization of user experiences, developing tailored content and responsive virtual assistants for specific tasks.

Generative AI makes our project more efficient, reduces costs, and drives innovation by automating repetitive tasks and streamlining workflows

**Process Flow**

Our mentor recommended that we follow these specific steps to complete the chatbot and fine-tuning process.The steps are as follow:

* git clone <https://github.com/intel/intel-extension-for-transformers.git>
* conda create -n itrex python-3.10 – y
* conda activate itrex
* pip install intel-extension-for-transformers
* cd /intel-extension-for-transformers/intel\_extension\_for\_transformers/neural\_chat/
* pip install -r requirements\_cpu.txt
* pip install -r requirements.txt
* huggingface-cli login
* python3 -m pip install jupyter ipykernel
* python3 -m ipykernel install --name neural-chat-1 –user

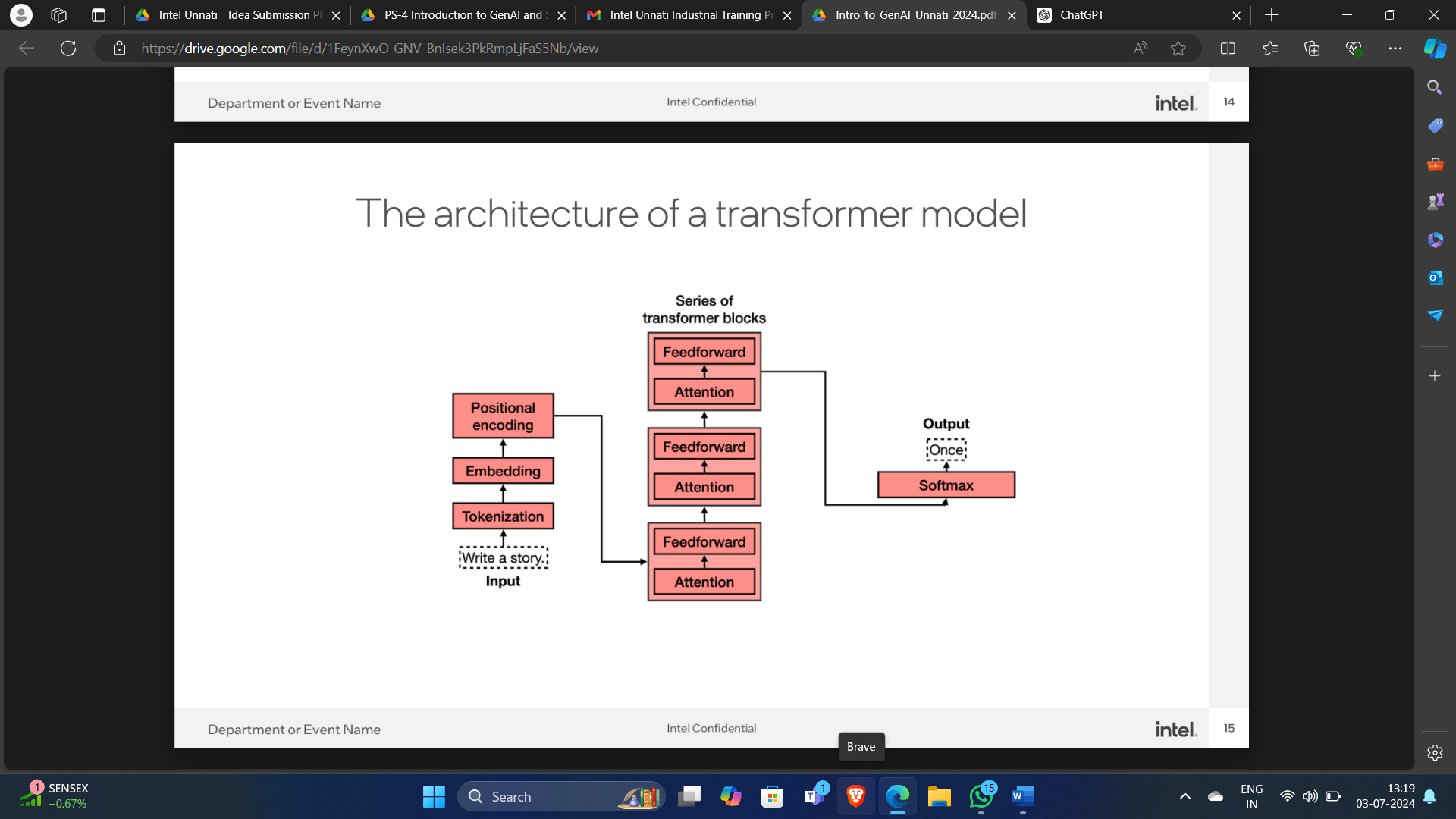
**Transformer**

A Transformer model consists of an encoder and a decoder. The encoder encodes the input sequence and passes it to the decoder which learns how to decode the representation for a relevant task.

The Transformer model is a powerful tool that helps computers understand and work with human language. It's particularly good at tasks like:

* Translating languages
* Summarizing long texts
* Generating new text
* Answering questions

**The architecture of a transformer model**



The architecture of a transformer model consists of several key components and steps:

* Input: Begins with the input text.
* Tokenization: Splits the input text into smaller units called tokens.
* Embedding: Translates tokens into a format the model can process.
* Positional Encoding: Provides positional information for each token in the sequence.
* Transformer Blocks:

1. Attention: Computes attention scores to focus on different parts of the sequence.
2. Feedforward: Processes representations further.

* Output: Uses the final processed data for predictions.
* Softmax: Converts the output into probabilities for generating the final output token.

**Technologies used**

These are technologies that we used for our project.

* Jupyter Notebooks
* Intel DevCloud for the Edge
* Hugging Face

Intel Developer Cloud provides a range of tools and resources that significantly support the development process, especially for tasks involving AI, machine learning, and high-performance computing. AI Tools from Intel:

* Data analytics
* Machine Learning
* Deep Learning
* Optimization and Inference

**Team Members and Contribution**

Team Name: DAV University-1

Team Members: 5

Applicant Name: Dilpreet Kaur

* Team Member: Nishant
* Team Member: Ashmeet Kaur
* Team Member: Sanjana Choudhary
* Team Member: Naman Sharma

The whole team worked together on this project's development. **Dilpreet** led the creation of the project report with the contributions from the team members. **Nishant** handled the setup of the initial Jupyter notebook and integrated the Hugging Face token. **Dilpreet** took charge of implementing the chatbot, testing prompts and **Ashmeet** helped in documenting the responses. **Sanjana** focused on fine-tuning the model and documenting the fine-tuning process. **Naman** was responsible for creating and organizing the GitHub repository.

**Conclusion**

The Intel® Unnati Industrial Training Program 2024 provided a valuable opportunity to explore Generative AI (GenAI) and Large Language Models (LLMs). By working on project "**Introduction to GenAI and Simple LLM Inference on CPU and Finetuning of the LLM Model in a Chatbot**" we got hands-on practice with how we can prepare our own chat box.

This training improves our knowledge on GenAI and how we can do model fine-tuning. The skills and knowledge acquired during this program have prepared us well for future AI projects. We are Thankful to Intel Unnati team for supporting us in learning process.