PS-16 Running GenAI on Intel AI Laptops and Simple LLM Inference on CPU and fine-tuning of LLM Models using Intel® OpenVINO™

The problem was to fine-tune a large language model (LLM) to generate hex color codes based on user prompts. This involved using a pre-existing dataset and model, and optimizing the process to ensure accurate color code generation.

Unique Idea Brief (Solution)

- The solution involved fine-tuning the ***TinyLlama-1.1B-Chat-v1.0*** model to understand and generate hex color codes from text descriptions. By leveraging the ***burkelibbey/colors*** dataset, the model was trained to map descriptive color names to their respective hex codes and and generate hex color codes from text descriptions. By leveraging the ***burkelibbey/colors*** dataset, the model was trained to map descriptive color names to their respective hex codes.

TinyLlama-1.1B-Chat-v1.0 :- This is the chat model finetuned on top of ***TinyLlama/TinyLlama-1.1B-intermediate-step-1431k-3T.*** The mode was initially fine-tuned on a variant of the ***UltraChat*** dataset, which contains a diverse range of synthetic dialogues generated by ChatGPT.

burkelibbey/colors :- This dataset is a collection of color descriptors paired with their respective hex color codes. It is designed to facilitate training models to understand and generate hex codes based on textual descriptions of colors.

Features Offered

Accurate Color Code Generation: The model accurately generates hex color codes from descriptive text inputs.

Optimized Performance: Utilization of efficient techniques like LoRA (Low-Rank Adaptation) and quantization to ensure the model performs well on hardware with limited resources.

User-Friendly Interface: The model can be integrated into various applications to provide real-time color code suggestions based on user input.

Process Flow

1. Data Preparation: Use the **burkelibbey/colors** dataset, which contains text descriptions and corresponding hex color codes.

2. Format Data: Convert the dataset into a format suitable for training the language model. This involves structuring the data into input-output pairs.

3. Load Pre-trained Model: Initialize the **TinyLlama-1.1B-Chat-v1.0** model with pre-trained weights.

4. Configure Tokenizer and Model: Set up the tokenizer and model configuration, including any quantization techniques to optimize performance.

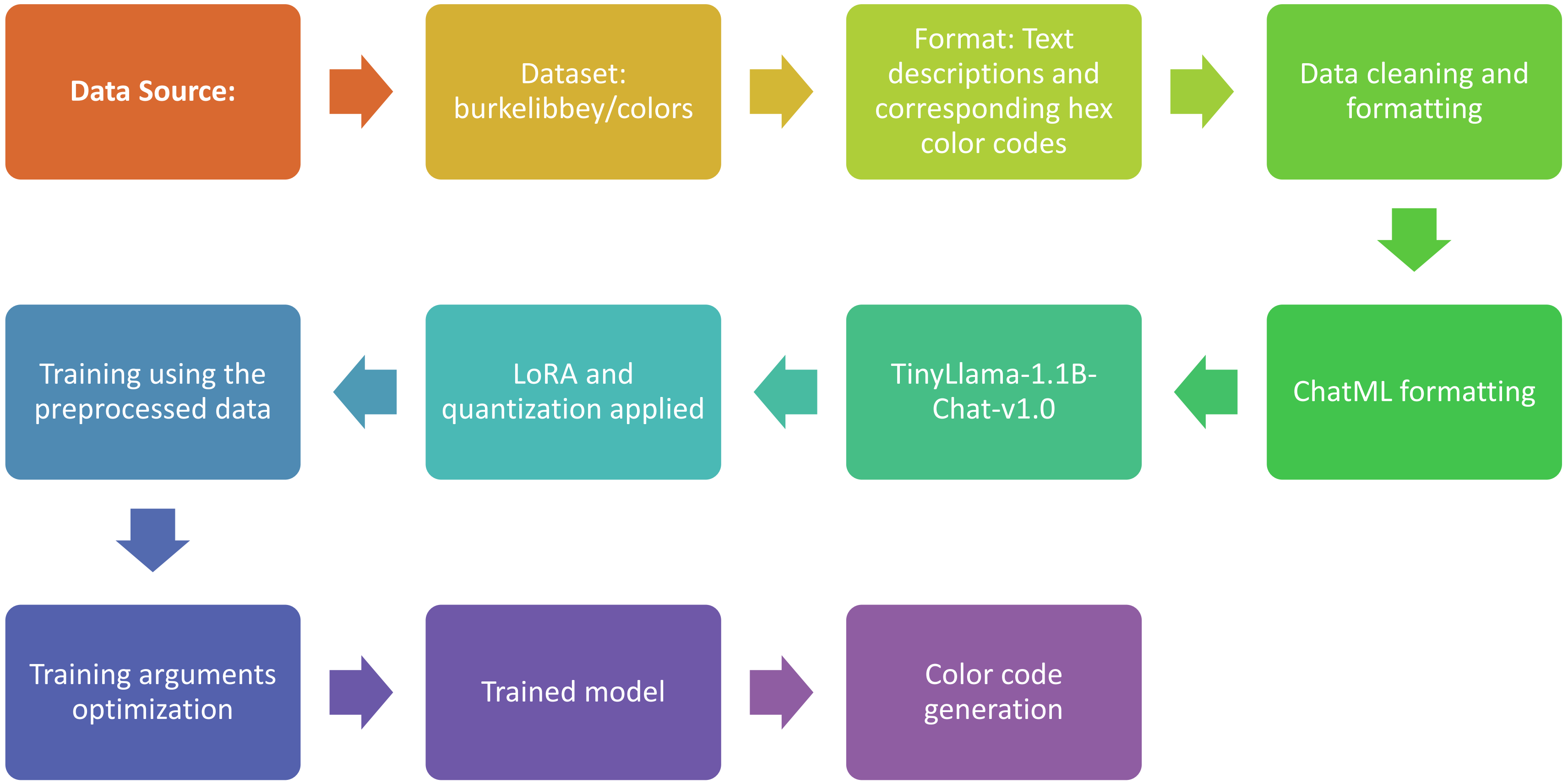
5. Setup Training Arguments: Define training parameters such as learning rate, batch size, and number of epochs.

6. Fine-Tune Model: Use the preprocessed dataset to fine-tune the model.

7. Evaluate Model Performance: Assess the model's ability to generate accurate hex color codes based on text inputs.

8. Save the Model: Storing the fine-tuned model for future use.

Architecture Diagram



Data Source:
Dataset: burkelibbey/colors
Format: Text descriptions and corresponding hex color codes
Data cleaning and formatting
ChatML formatting
TinyLlama-1.1B-Chat-v1.0
LoRA and quantization applied
Training using the preprocessed data
Training arguments optimization
Trained model
Color code generation

Technologies Used

Transformers

Datasets

BitsAndBytes

Peft

TRL

Team Members and Contributions

- Vivek Karadbhajne (Team Leader)
  - Model Development and fine-tuning and Directing
- Sanskar Sawane
  - Model and OpenVINO integration
- Vedant Patil
  - Integration and UI development

intel®