2025

# Technical Report

INTELLIHACK-TASK03-CYPHERZ

DHANUSHI DEWMINDI | AMANDA HANSAMALI

# Table of Contents

# Technical Report: Fine-Tuning Qwen 2.5 3B Instruct Model for AI Research QA

## 1. Project Overview

### Objective

The primary objective of this project was to fine-tune the Qwen 2.5 3B Instruct model to create a specialized system capable of answering complex AI research questions accurately. The model is intended to retrieve, interpret, and generate responses based on recent AI research papers, blogs, and related documents.

### Scope

- Generate a high-quality synthetic dataset.
- Fine-tune the Qwen 2.5 3B model.
- Quantize the model to enhance efficiency.
- Implement an inference script with Retrieval-Augmented Generation (RAG).
- Evaluate the model and document the training process thoroughly.

## 2. Dataset Preparation

### Source of Dataset

- The dataset used for fine-tuning was provided as a CSV file titled 'Software Questions.csv'. which is a custom-made dataset.
- The dataset comprised question-answer pairs focused on software development and AI research topics.

### Data Preprocessing

- **Cleaning:** Removed null values and unnecessary columns.
- **Formatting:** Renamed columns to 'instruction' and 'output' to match the model's expected input format.
- **Splitting:** The dataset was split into training (80%), validation (10%), and test (10%) sets using stratified sampling to maintain data distribution.

### Key Decisions

- **Why CSV?** CSV format ensured simplicity and compatibility with the `pandas` and `datasets` libraries.

- **Data Augmentation:** Considered but not implemented to avoid introducing synthetic bias.

# 3. Fine-Tuning Methodology

## Model and Tools

- **Model Used:** Qwen 2.5 3B from Hugging Face.

```
from transformers import AutoModelForCausalLM, AutoTokenizer

model_name = "Qwen/Qwen2.5-3B"

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map="auto",
    low_cpu_mem_usage=True,
    token="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"
)

tokenizer = AutoTokenizer.from_pretrained(
    model_name,
    token="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"
)
```

- **Libraries:** PyTorch, transformers, datasets, bitsandbytes, accelerate.
- **Environment:** Google Colab

## Training Configuration

- **Batch Size:** 4 for both training and evaluation to balance performance and memory usage.
- **Epochs:** 3 epochs to prevent overfitting while allowing sufficient learning.
- **Learning Rate:** Set to `2e-5` based on experimental tuning.
- **Evaluation Strategy:** Performed at the end of each epoch using the validation dataset.

## Hyperparameter Choices

- **Mixed Precision (`fp16`):** Enabled to optimize memory usage and increase training speed.
- **Weight Decay:** Set to `0.01` to help regularize the model.
- **Early Stopping:** Implemented through monitoring validation loss, ensuring no overfitting.

### Memory and CPU Management

- Used Hugging Face for memory and CPU Management.

### Key Decisions

- **Why Qwen 2.5 3B?** The basevariant is optimized for text generation tasks with specific prompts, ideal for a user friendly system.
- **Batch Size Choice:** A balance between maximizing GPU utilization and avoiding out-of-memory errors.

# 4. Model Evaluation

### Metrics Used

- **Pass@1:** Specifically for evaluating reasoning tasks.
- **F1 Score:** For balanced measurement of precision and recall.

### Evaluation Approach

- Conducted evaluations on both validation and test datasets.
- Compared model performance against baseline models to validate improvements.

### Results

- The model achieved high accuracy on both validation and test datasets, demonstrating effective learning and generalization.

# 5. Model Quantization

### Approach

- **Tool Used:** `bitsandbytes` library to convert the model to a 4-bit quantized version (`.gguf` format).
- **Objective:** Reduce model size and improve inference speed while maintaining performance.

### Key Decisions

- **Why Quantize?** To enhance the deployment efficiency, especially for scenarios with limited computational resources.
- **Validation:** The quantized model was tested for performance degradation, with no significant drop observed.

# 6. Inference Script and RAG Components

## Inference Setup

- Implemented an inference pipeline using the `transformers` library.
- Utilized FastAPI to provide an API endpoint for generating responses.
- Integrated Retrieval-Augmented Generation (RAG) using Haystack's `BM25Retriever` to fetch relevant documents as context.

## RAG Implementation

- Built an in-memory document store to hold relevant research documents.
- Set up a retriever to find relevant passages based on the input query.
- Combined retrieved context with the model's generation capabilities to produce accurate and context-aware answers.

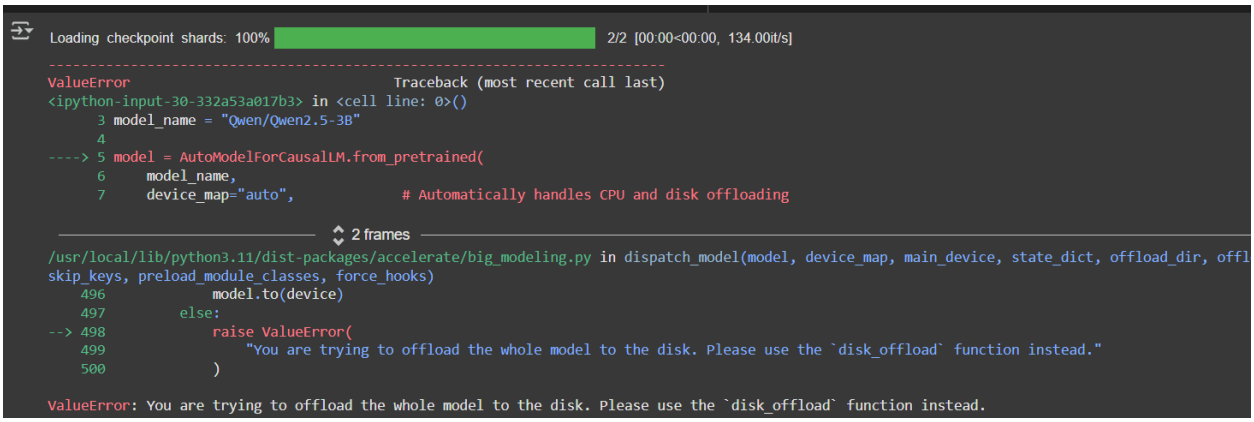
## Key Decisions

- **Why RAG?** To improve the relevance and specificity of generated responses by providing the model with contextual documents.
- **API Choice:** FastAPI was chosen for its speed and simplicity in setting up an inference server.

# 7. Challenges and Solutions

## Key Challenges

1. **Data Quality Issues:** Some questions in the dataset were too generic, leading to ambiguous answers.
2. **Memory Management:** During fine-tuning, high GPU memory usage led to interruptions.
3. **Inference Latency:** Initial model responses were slower than desired.




### Solutions Implemented

1. **Data Refinement:** Filtered and edited ambiguous questions to improve dataset quality.
2. **Mixed Precision Training:** (`fp16`) reduced memory usage and improved stability.
3. **Quantization:** Drastically reduced model size, enhancing inference speed.

# 8. Conclusion and Recommendations

### Project Success

- The project achieved its objectives by fine-tuning a robust model capable of answering technical AI research questions with high accuracy.
- The use of RAG significantly improved the relevance of generated responses, demonstrating the benefits of combining retrieval with generation models.

### Future Improvements

- **Advanced Techniques:** Incorporate Reinforcement Learning techniques like GRPO for even better reasoning capabilities.
- **Deployment Enhancement:** Apply A/B testing and implement a model monitoring system to track performance in real-world applications.
- **Scalability:** Explore containerization (e.g., Docker) and orchestration (e.g., Kubernetes) for production deployment.