AI Model Weekend Hackathon
Fine-Tuning DistilBERT for Sentiment Analysis

AI-Assisted Learning

September 20, 2025

1 Event Overview

The AI Model Weekend Hackathon is organized by TakenoLAB ICT Academy Tech Hub, a local firm dedicated to bridging the digital divide and empowering underserved communities through technology and innovation, particularly at the Dzaleka Refugee Camp in Malawi.[1] [2]

1.1 Objectives

1. **Skill Development**: Hands-on experience in LLM fine-tuning, AI-powered automation, and AI native applications.
2. **Portfolio Building**: Complete a working project you can showcase.
3. **Networking**: Connect with AI practitioners and enthusiasts in Malawi.
4. **Problem Solving**: Apply AI techniques to real challenges.

1.2 Requirements

- Basic computer usage skills
- Python skills
- Passion about AI and Technology
- Register and show up if selected
- Limited to 12 participants only
- TakenoLAB provides computers and full internet access

1.3 Application and Event Details

Apply to: info@takenolab.org
Application period: From Monday 15th to Thursday 18th Sept. 2025
Event date: Saturday 20th Sept. 2025

[1] `https://takenolab.org/`
[2] `https://mw.linkedin.com/company/takenolab`

2 Mentor

The mentor for this hackathon is Remy Gakwaya, who holds a BA in Business Administration and is a Software Engineer. He is the Founder and CEO of TakenoLAB and RELON Malawi, an AI and Software Engineer with a background in teaching coding and entrepreneurial skills to refugees and underserved communities.[3] [4] He has collaborated with organizations like UNHCR and Microsoft to provide programming education.[5]

3 Introduction to the Lesson

Today's lesson, part of the AI Model Weekend Hackathon, focused on fine-tuning the DistilBERT model for sentiment analysis using the IMDb dataset. The process involved setting up the environment, preparing the dataset, configuring the model with LoRA adapters, training, evaluating, and saving the model. Below are the key points and steps covered.

4 Environment Setup

We began by installing necessary Python libraries for fine-tuning, including:

- `bitsandbytes` and `accelerate` for efficient training.
- `xformers` for optimized transformer operations.
- `peft`, `trl`, and `unsloth` for parameter-efficient fine-tuning.
- `transformers==4.55.4` and `trl==0.22.2` for model and training utilities.

The installation commands dynamically adjusted the `xformers` version based on the PyTorch version.

5 Library Imports

We imported essential libraries for the task:

- `unsloth.FastLanguageModel`: For loading and fine-tuning the model.
- `transformers.AutoModelForSequenceClassification`: For sequence classification tasks.
- `datasets`: To load and process the IMDb dataset.
- `sklearn.metrics.accuracy_score`: To compute evaluation metrics.

[3] `https://www.linkedin.com/in/gakwayaremy`

[4] `https://gem.snhu.edu/2021/11/12/snhu-gem-alums-takenolab-technology-school-grows-and-gains-recog`

[5] `https://www.unhcr.org/us/news/stories/refugees-learn-code-new-future-malawi`

6 Model and Dataset Preparation

6.1 Model Initialization

We used `distilbert/distilbert-base-uncased` as the base model, configured for binary classification (positive/negative sentiment) with:

- `num_labels=2` for binary classification.
- `max_seq_length=512` to match DistilBERT's input size.
- `load_in_4bit=True` for memory-efficient 4-bit quantization.

LoRA adapters were added with:

- `r=16` (low-rank adaptation rank).
- `target_modules=["q_lin", "v_lin"]` for DistilBERT-specific layers.
- `lora_alpha=16` and `lora_dropout=0` for adapter configuration.

6.2 Dataset Preparation

We loaded a subset of the IMDb dataset (`train[:1000]`) and processed it:

- Formatted examples to include `text` and `labels`.
- Tokenized the dataset using the DistilBERT tokenizer with `max_length=512`, `padding`, and `truncation`.
- Removed unnecessary columns (`text`, `label`) post-tokenization.

7 Training Configuration

The `Trainer` was set up with `TrainingArguments`:

- `per_device_train_batch_size=32` for batch processing.
- `num_train_epochs=1` for a single epoch.
- `learning_rate=5e-5` and `optim="adamw_8bit"` for optimization.
- Mixed precision training with `fp16` or `bf16` based on hardware support.
- `output_dir="outputs"` for saving training results.

A `compute_metrics` function was defined to calculate accuracy using `accuracy_score`.

8 Training

The model was trained using the `trainer.train()` method. Training statistics were stored in `trainer_stats`.

9 Evaluation and Testing

9.1 Quick Test

A simple sentiment analysis test was performed using `transformers.pipeline`:

```
sentence1 = "This movie was not that great at all."
classifier = pipeline("sentiment-analysis", model=model, tokenizer=
    tokenizer)
classifier(sentence1)
```

9.2 Deep Testing

The test split of the IMDb dataset was loaded, tokenized, and evaluated using a separate `Trainer` instance. Results were printed as:

```
eval_results = eval_trainer.evaluate()
print(f"Evaluation results: {eval_results}")
```

9.3 Review-Based Testing

Two sample reviews were tested:

- Positive: "This movie was an absolute masterpiece! The acting, direction, and story were perfect."
- Negative: "I was incredibly disappointed by this movie. The plot was boring and the ending made no sense."

The model predicted sentiments with confidence scores using softmax on logits.

10 Saving the Model

The model and tokenizer were saved locally to `sentim_movies` and uploaded to Hugging Face:

```
model.save_pretrained("sentim_movies")
tokenizer.save_pretrained("sentim_movies")
model.push_to_hub("aired/sentim_movies")
tokenizer.push_to_hub("aired/sentim_movies")
```

Hugging Face login was performed using `notebook_login()`.