**1. Project Idea**

This project focuses on legal case summarization. The goal is to automatically generate concise and coherent summaries from complex legal judgments using a fine-tuned Large Language Model (LLM). The model assists legal professionals and researchers by reducing reading time and enhancing understanding of lengthy case texts.

**2. Dataset Information**

The dataset used for fine-tuning is from Hugging Face:

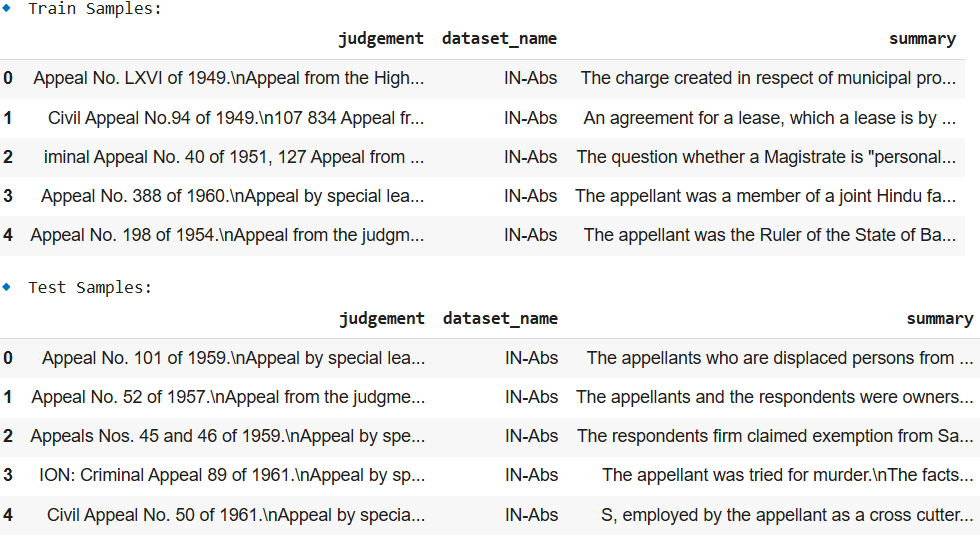
* **Source:** [joelniklaus/legal\_case\_document\_summarization](https://huggingface.co/datasets/joelniklaus/legal_case_document_summarization)
* **Link:**https://huggingface.co/datasets/joelniklaus/legal\_case\_document\_summarization
* **Format:** DatasetDict
* **Splits:**
  + **Train:** 7,773 examples
  + **Test:** 200 examples
* **Features:**
  + judgement: The full legal case text.
  + summary: The human-written summary.
  + dataset\_name: Source dataset identifier.
* **Preprocessing:**

1)Truncated both judgement and summary to a maximum of 1000 words.

2)Cleaned text (removed headers, special characters if needed)

3)Tokenized using FLAN-T5 tokenizer.

**This dataset focuses on summarizing long legal judgments into concise summaries using natural language generation.**



A screenshot of a computer

AI-generated content may be incorrect.

**3. Base Model Information**

* **google/flan-t5-base**

**Architecture:**

* **T5 (Text-to-Text Transfer Transformer):** This architecture is based on the Transformer model and is designed to treat every NLP task as a text-to-text problem. For example, text classification is treated as generating a label (text) from an input (text), and machine translation is treated as generating a translated sentence from the input sentence.
* **Size:** base typically has 220 million parameters, which is a mid-sized version of the T5 model. It balances performance and computational efficiency.
* **Layers:** The base version has 12 Transformer layers.
* **Hidden Units:** 768 hidden units.
* **Attention Heads:** 12 attention heads per layer.

**FLAN Fine-Tuning:**

* **FLAN (Fine-tuned Language Net):** FLAN refers to a method where models are further fine-tuned on a diverse set of tasks to improve their generalization capabilities across different natural language understanding and generation tasks. This includes tasks like question answering, summarization, and reasoning.
* **Supervised Fine-Tuning:** The FLAN-T5 models are fine-tuned on a collection of datasets from various tasks to enhance their performance beyond standard pre-training.

**Capabilities:**

* **Text Generation:** The model is capable of generating coherent and contextually relevant text based on input prompts.
* **Question Answering (QA):** It can answer questions based on given contexts (e.g., paragraphs, articles).
* **Summarization:** It can generate concise summaries from longer texts, making it useful for tasks like document summarization.
* **Translation:** It can handle language translation tasks.
* **Text Classification:** It can be used for tasks like sentiment analysis, spam detection, or other classification tasks when fine-tuned for such purposes.
* **Text-based Inference:** It is capable of performing logical reasoning on text to derive conclusions or make inferences.

**Use Cases:**

* Summarization.
* Generating responses to prompts, such as chatbot functionalities.
* Performing tasks like paraphrasing or expanding text.
* Language translation and multilingual tasks.

**4. Brief Explanation of Fine-Tuning with LoRA**

**Low-Rank Adaptation (LoRA)** is a technique designed to efficiently fine-tune pre-trained models, especially large models, by introducing trainable low-rank matrices into specific parts of the model. This approach reduces the computational and memory overhead typically associated with fine-tuning large models.

**High-Level Overview of LoRA:**

LoRA is a method where instead of updating the entire pre-trained model during fine-tuning, you add small learnable low-rank matrices to the model's weight matrices. These additional matrices (called **LoRA layers**) are trained while keeping the original model weights frozen. This leads to significantly fewer parameters being updated during training, making the fine-tuning process more memory- and compute-efficient.

**Key Concepts:**

1. **Low-Rank Matrices:**
   * LoRA introduces low-rank matrices to the attention layers of the model. In your case, it is applied to the **query (q)** and **value (v)** modules of the attention mechanism.
   * These matrices are much smaller in size compared to the original weight matrices, making them computationally inexpensive to optimize.
2. **Efficient Fine-Tuning:**
   * Instead of updating the entire model, LoRA only updates the additional low-rank matrices. This reduces the number of parameters that need to be trained, speeding up the training process while still achieving performance gains.
3. **Task-Specific Adaptation:**
   * LoRA allows for task-specific fine-tuning by adjusting only a small subset of the model parameters, ensuring that the original pre-trained model is not overfit and retaining its broad generalization capabilities.
4. **LoRA Configuration (LoraConfig):**
   * r=32: This is the rank of the low-rank matrices. A higher value would result in more parameters being adapted.
   * lora\_alpha=32: A scaling factor for the LoRA matrices. It controls the strength of the adaptation applied to the model.
   * target\_modules=["q", "v"]: LoRA is applied to the query and value components of the attention mechanism in the model, allowing for more efficient adaptation of these crucial parts of the transformer architecture.
   * lora\_dropout=0.05: A small dropout applied to the LoRA matrices to prevent overfitting.
   * bias="none": No bias term is applied to the LoRA matrices.
   * task\_type=TaskType.SEQ\_2\_SEQ\_LM: Specifies the task type (sequence-to-sequence language modeling) for fine-tuning.
5. **Training Arguments (TrainingArguments):**
   * output\_dir: Specifies where the fine-tuned model will be saved.
   * auto\_find\_batch\_size=True: The batch size will be automatically adjusted based on the available memory.
   * learning\_rate=1e-3: A higher learning rate is used for LoRA fine-tuning compared to full fine-tuning, as LoRA updates only a small portion of the model.
   * num\_train\_epochs=50: The model will be trained for 50 epochs.
   * max\_steps=500: The total number of steps is capped at 500, limiting the training duration.
   * report\_to="none": No external reporting during training (e.g., no logs sent to external services).
6. **Trainer (Trainer):**
   * The Trainer class is used to manage the training process. It takes the following inputs:
     + model=peft\_model: This is the model that has been adapted with LoRA.
     + train\_dataset=tokenized\_datasets["train"]: The training data (tokenized text data).
     + args=peft\_training\_args: The training configuration, including learning rate, number of epochs, etc.

**Why LoRA is Used:**

* **Efficiency:** Fine-tuning large models like T5 can be computationally expensive. LoRA significantly reduces the number of parameters that need to be updated, making it faster and less resource-intensive.
* **Flexibility:** By only fine-tuning specific parts of the model (e.g., attention layers), LoRA can adapt the model to specific tasks without requiring full re-training.
* **Preservation of Generalization:** LoRA ensures that the pre-trained model's generalization capabilities are preserved, as it doesn't change the original weights, just adds efficient adaptations.

In summary, LoRA allows you to fine-tune a large pre-trained model like flan-t5-base in a computationally efficient way, targeting specific layers (e.g., attention layers) and adding small, task-specific low-rank matrices. This method enables quick adaptation to new tasks while maintaining the generalization power of the base model. **5. Evaluation Methods**

A graph of different colored bars

AI-generated content may be incorrect.

**6. Structured Output**

INPUT PROMPT:

Summarize the following conversation.

“The trial court improperly instructed the jury that it should not consider

the victim's delayed reporting when evaluating her credibility because,

although the court's instruction was proper at the time that it was given,

the Supreme Court's subsequent decision in State v. Adam P. (351 Conn.

213), which was released while the defendant’s appeal was pending before

this court, reinstated the standard, articulated in State v. Troupe, (237 Conn.

284), that a defendant in a sexual assault case is entitled to an instruction

that any delay by the victim in reporting the incident is a matter for the

jury to consider in evaluating the weight of the victim's testimony.”

Summary:

-------------------------------------------------------------------------------------

ORIGINAL MODEL:

The jury court improperly instructed the jury that it should not consider the delay by the victim in reporting the incident. The court's instruction was proper at the time that it was given. The Supreme Court's subsequent decision in State v. Troupe, (State v.)

-------------------------------------------------------------------------------------

PEFT MODEL:

The jury was instructed to consider the delay by the victim in reporting the incident as a matter for the jury to consider in evaluating the weight of the victim's testimony.