Dataset Creation

To address the challenge of acquiring real depositions for fine-tuning Mistral-7B-Instruct-v0.2 for automated summarization, we can leverage large language models (LLMs) like Gemini or ChatGPT to generate synthetic depositions.
Here's a step-by-step guide:

1. Generating Synthetic Depositions:

- Prompt the LLM: Provide clear instructions to the LLM, including the type of deposition (e.g., accident, theft) and any relevant details (e.g., parties involved, timeframes).
- Iterative Refinement: Start with a basic prompt and progressively refine the deposition by prompting the LLM to "extend the deposition based on the previous response."
- Summarization: Once satisfied with the deposition, prompt the LLM to "generate a summary of the deposition."

2. Building the Dataset:

- Repeat the process: Generate multiple deposition-summary pairs by repeating steps ensuring diverse scenarios in the dataset.

Building a Compatible Dataset for Fine-Tuning

After generating synthetic depositions and summaries using an LLM, we need to structure the data appropriately for fine-tuning Mistral-7B-Instruct-v0.2. Here's how:

1. **Create a CSV File:**
    a. Open a spreadsheet editor or text editor.
    b. Create two columns titled "Deposition" and "Summary."
    c. Fill each row with a corresponding deposition-summary pair.
    d. Use the pipe symbol ("|") to separate the deposition from its summary within each row. This adheres to standard formatting for many data processing tasks.
2. **Understand Mistral AI's Instruction Format:**
    a. Mistral AI requires a specific format to fine-tune their model effectively. This involves surrounding your instructions (in our case, the depositions) with special tokens:
        i. [INST]: indicates the beginning of an instruction.
        ii. [/INST]: marks the end of an instruction

```
Example:

text = "<s>[INST] What is your favourite condiment? [/INST]"

"Well, I'm quite partial to a good squeeze of fresh lemon juice.
It adds just the right amount of zesty flavour to whatever I'm
cooking up in the kitchen!</s> "
```

Our dataset will consist of three main elements:

- **Deposition:** This field contains the raw text of the legal statement.
- **Summary:** This field holds the condensed version of the corresponding deposition.
- **Text:** This crucial field is specifically formatted for the fine-tuning process, as it guides the model during training. Its structure will be further explained later in this document.

Key Points and Logic:

JSONL Format: Each row in the JSON Lines file will be a self-contained JSON object, representing a single training example for fine-tuning the Mistral-7B-Instruct-v0.2 model.

Data Elements: Each row will have the following structure:

JSON

{

"deposition": "(the text of the deposition)",

"summary": "(the corresponding summary)",

"text": "(the formatted text ready for Mistral-7B-Instruct-v0.2)"

}

We will utilize a function responsible for concatenating the instructions, deposition, summary, and special tokens according to Mistral AI's format. The output will populate the text key of each row.

Example:
Let's assume you have the following dummy deposition-summary pair in your CSV file:

- **Deposition:** "The witness saw a blue car speeding through the intersection and hitting a pedestrian on the crosswalk."
- **Summary:** "Blue car runs intersection, hits pedestrian."

JSONL Row: After formatting by the code, the corresponding row in your JSONL file would look like:

JSON

```
{
    "deposition": "The witness saw a blue car speeding through the
intersection and hitting a pedestrian on the crosswalk.",
    "summary":  "Blue car runs intersection, hits pedestrian.",
    "text": "<s>[INST] Provide a summary of the following: The witness saw a
blue car speeding through the intersection and hitting a pedestrian on the
crosswalk. [/INST] \n Blue car runs intersection, hits pedestrian. </s>"
}
```

Explanation:

deposition and summary: These fields contain the raw deposition and summary text, respectively.

text: This field contains the deposition, your instruction ("Provide a summary of the following"), and the summary, surrounded by the [INST], [/INST] tokens. This is what Mistral AI's model will be trained on.

How the model uses this:

Mistral-7B-Instruct-v0.2 is designed to learn from examples. When you feed it a JSONL file containing examples like this, it internalizes the pattern:

[INST] indicates the start of a task

Text following [INST] is the input to process

Text following \n is the correct solution

[/INST] signals the end of the task

The samples are divided into 3 categories of the dataset as explained below:

Training Dataset

- **Purpose:** The training dataset is used to train the machine learning model. During training, the model learns patterns and relationships in the data, optimizing its parameters to minimize the error between predicted and actual values.
- **Composition:** The training dataset typically constitutes the largest portion of the available data.
- **Usage:** The model iteratively processes batches of data from the training dataset, updating its parameters through techniques like gradient descent or backpropagation.
- **Outcome:** After training, the model should be capable of making predictions on new, unseen data.

Validation Dataset

- **Purpose:** The validation dataset is used to tune hyperparameters and assess the model's performance during training.
- **Composition:** The validation dataset is a separate portion of the dataset that is not used for training. It is often a subset of the overall dataset.
- **Usage:** During the training process, the model's performance on the validation dataset is periodically evaluated. This evaluation helps identify potential overfitting and guides adjustments to the model's architecture or hyperparameters.
- **Outcome:** By monitoring the model's performance on the validation dataset, practitioners can make informed decisions to improve the model's generalization capabilities.

Test Dataset

- **Purpose:** The test dataset is used to evaluate the final performance of the trained model.
- **Composition:** Like the validation dataset, the test dataset is separate from both the training and validation datasets.
- **Usage:** Once the model has been trained and tuned using the training and validation datasets, it is evaluated on the test dataset. The test dataset provides an unbiased assessment of the model's ability to generalize to unseen data.
- **Outcome:** The performance metrics obtained from the test dataset reflect the model's performance in real-world scenarios. These metrics guide decisions regarding model deployment and further improvements.