The Llama fine-tuning libraries

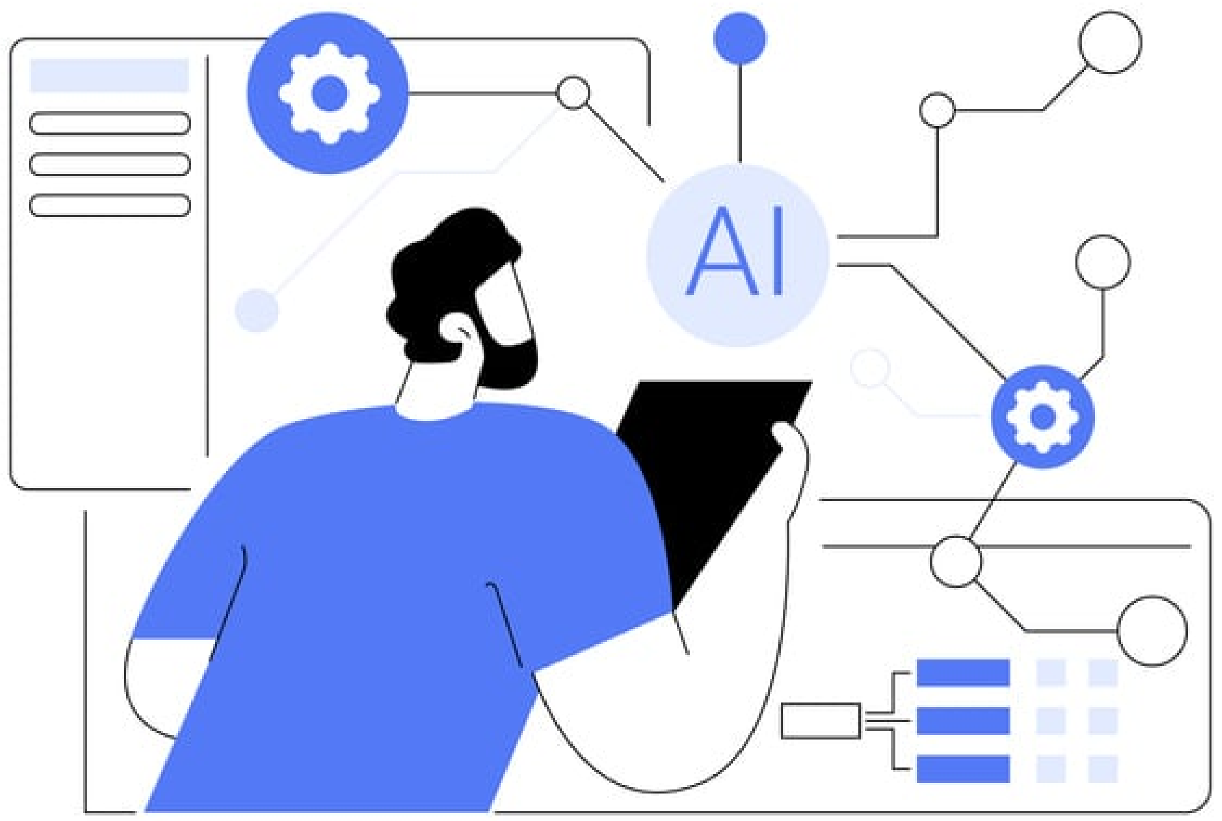
FINE-TUNING WITH LLAMA 3

Francesca Donadoni
Curriculum Manager, DataCamp

When to use fine-tuning

- Pre-trained model
- Uses specialized data

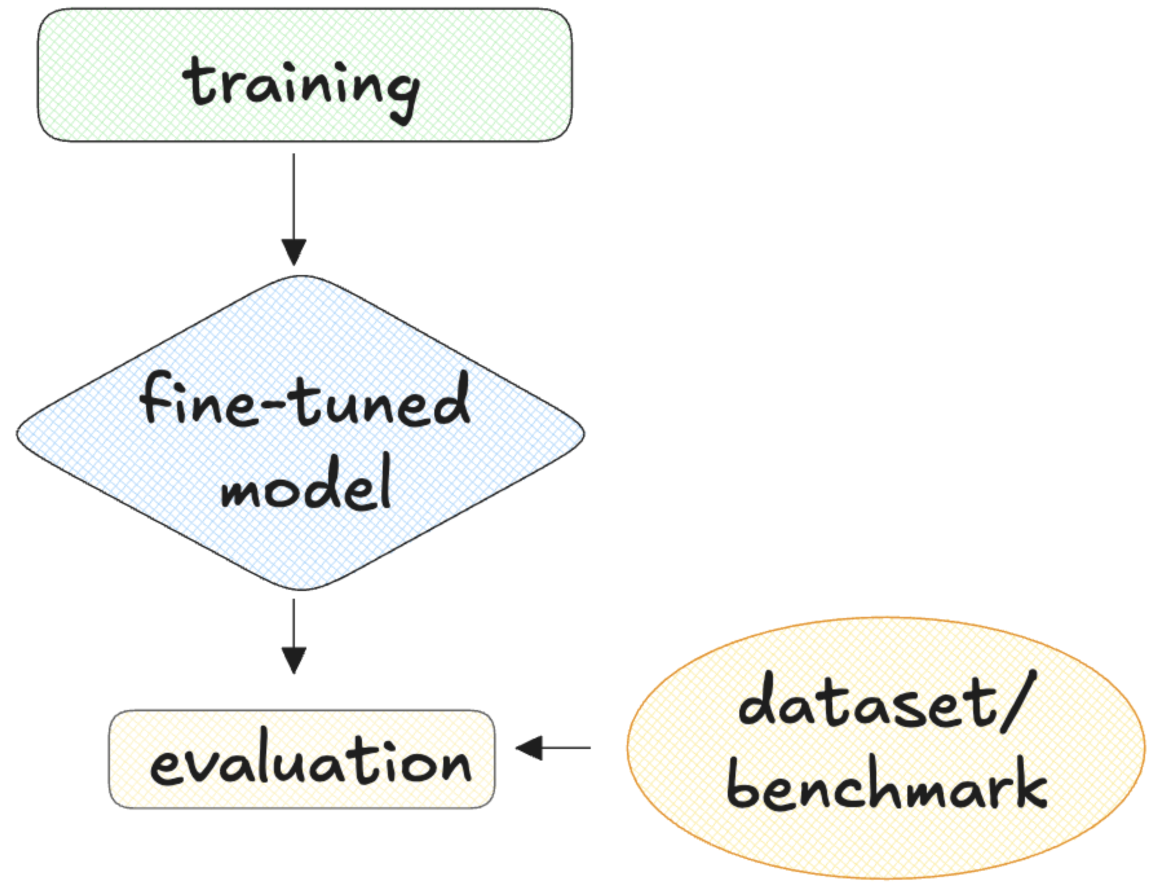
- Improve accuracy
- Reduce bias
- Improve knowledge base



How to use fine-tuning

- Quality of the data
- Model's capacity
- Task definition

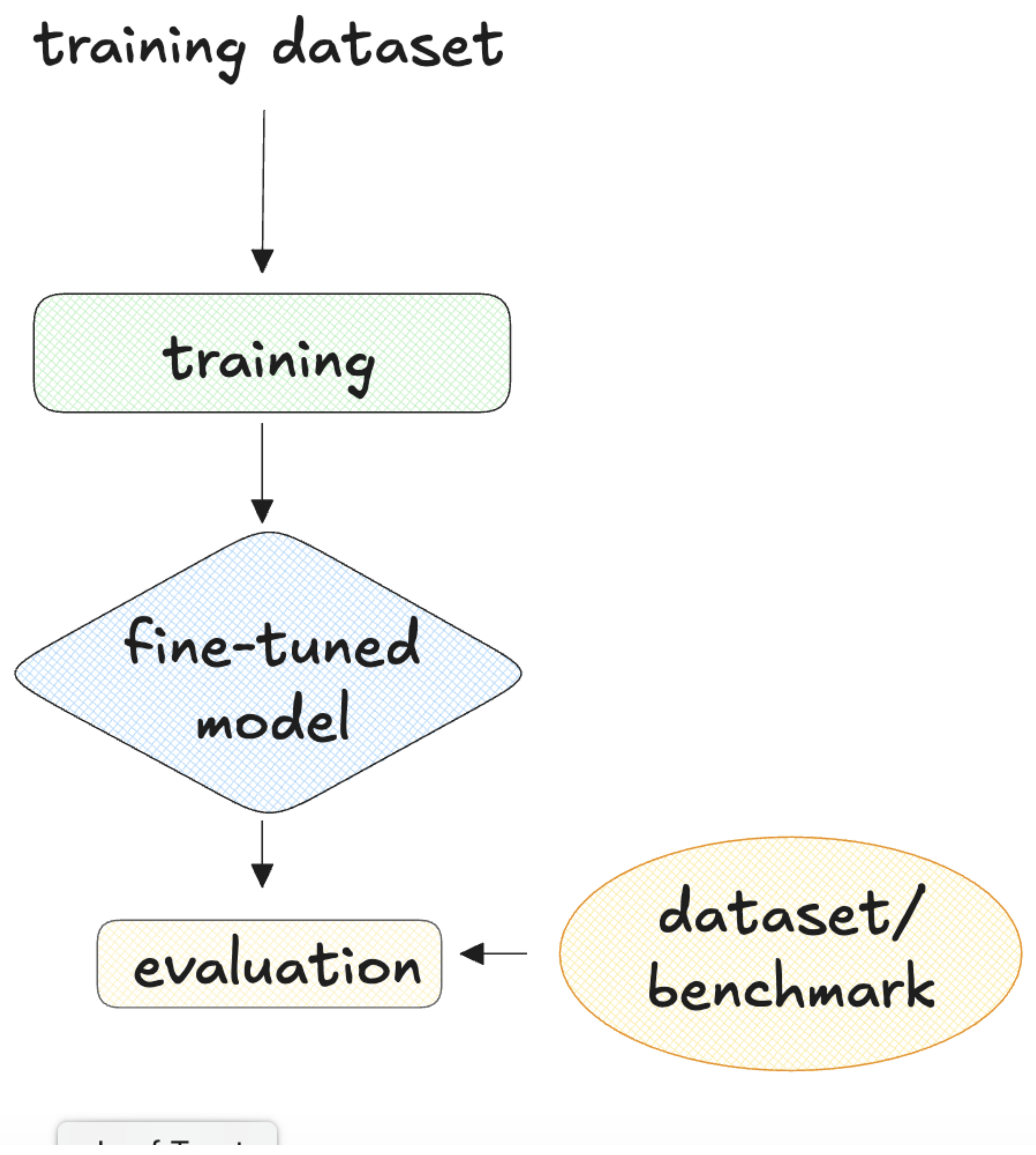
- Fine-tuning process



How to use fine-tuning

- Quality of the data
- Model's capacity
- Task definition

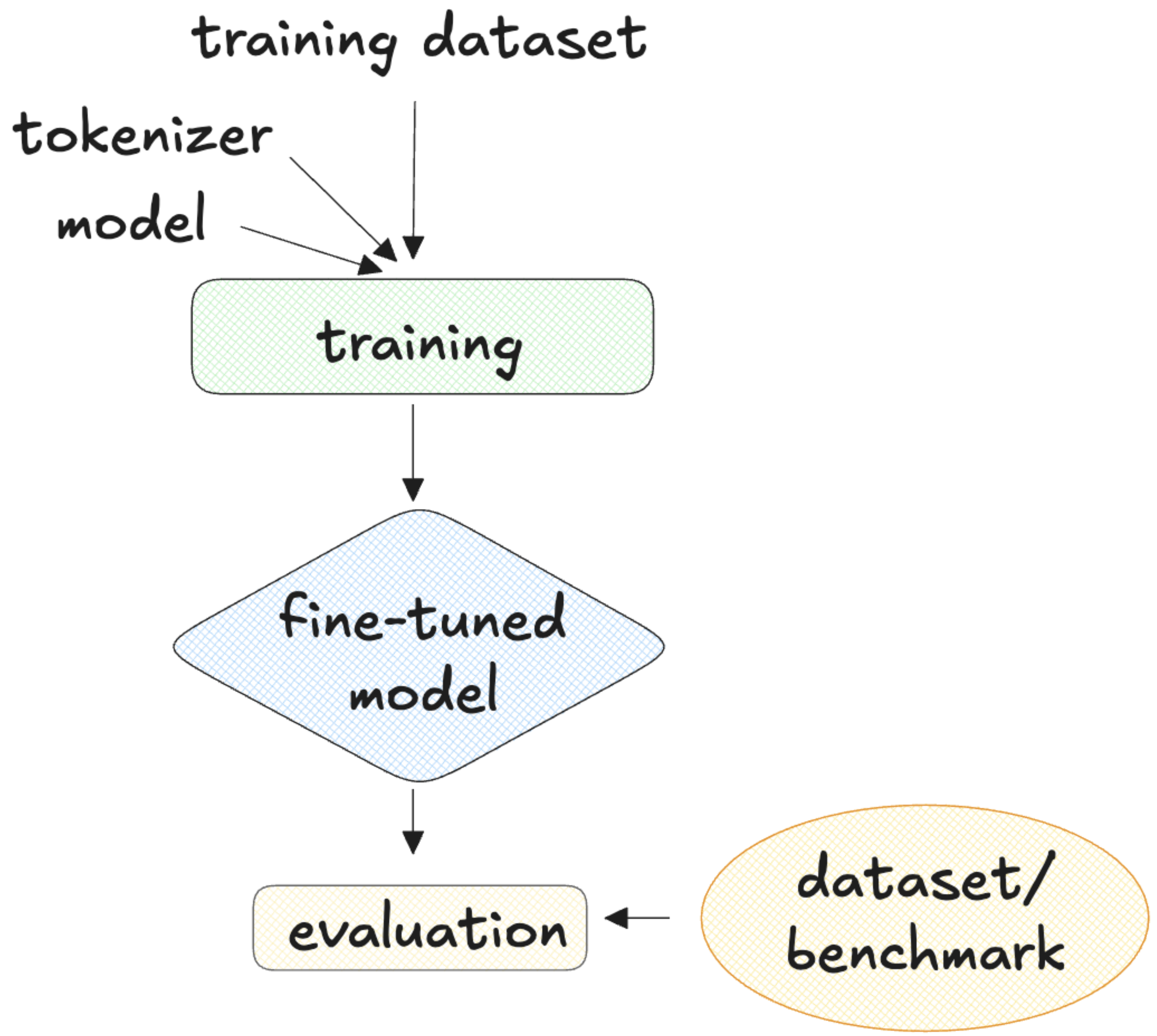
- Fine-tuning process



How to use fine-tuning

- Quality of the data
- Model's capacity
- Task definition

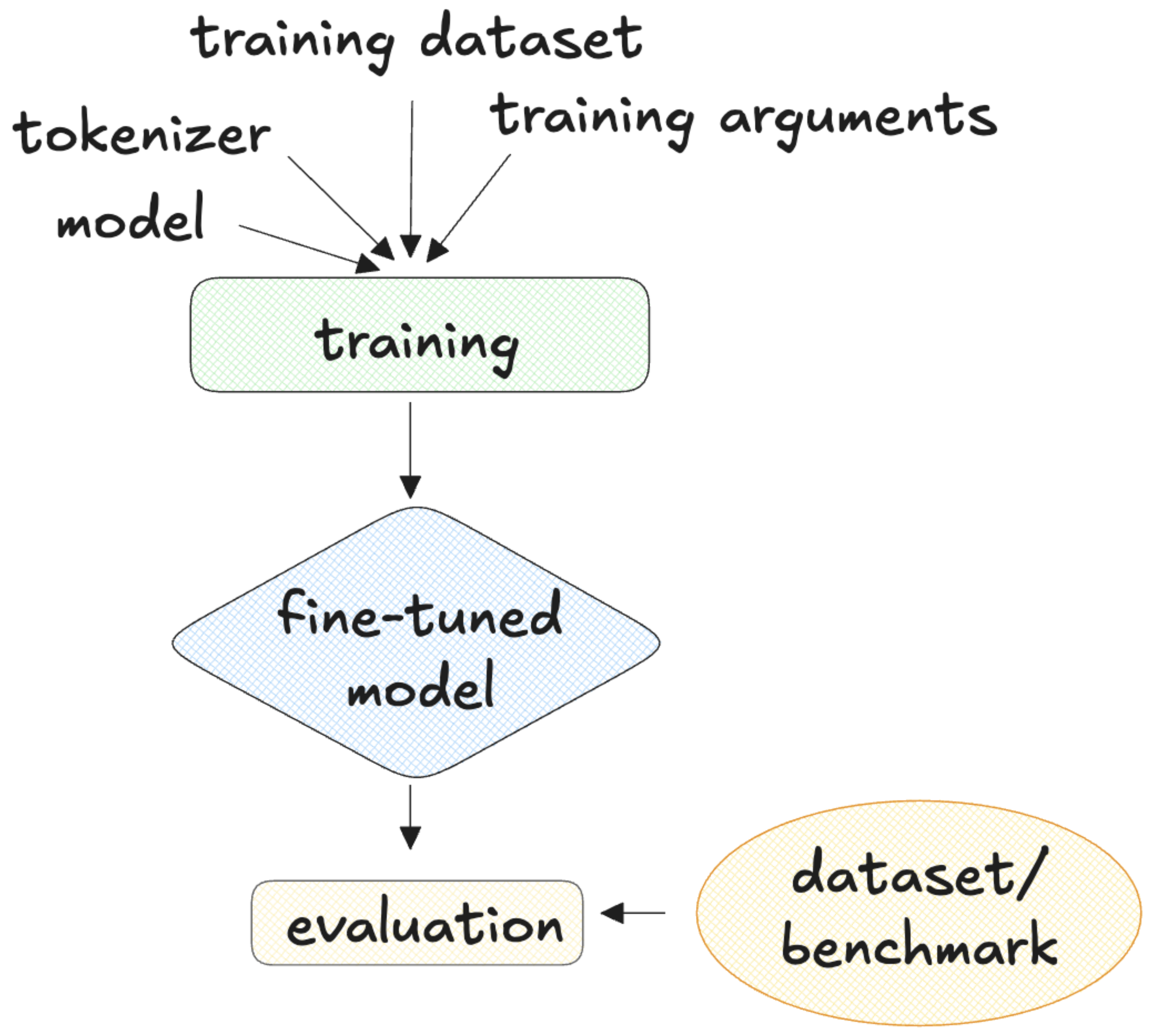
- Fine-tuning process



How to use fine-tuning

- Quality of the data
- Model's capacity
- Task definition

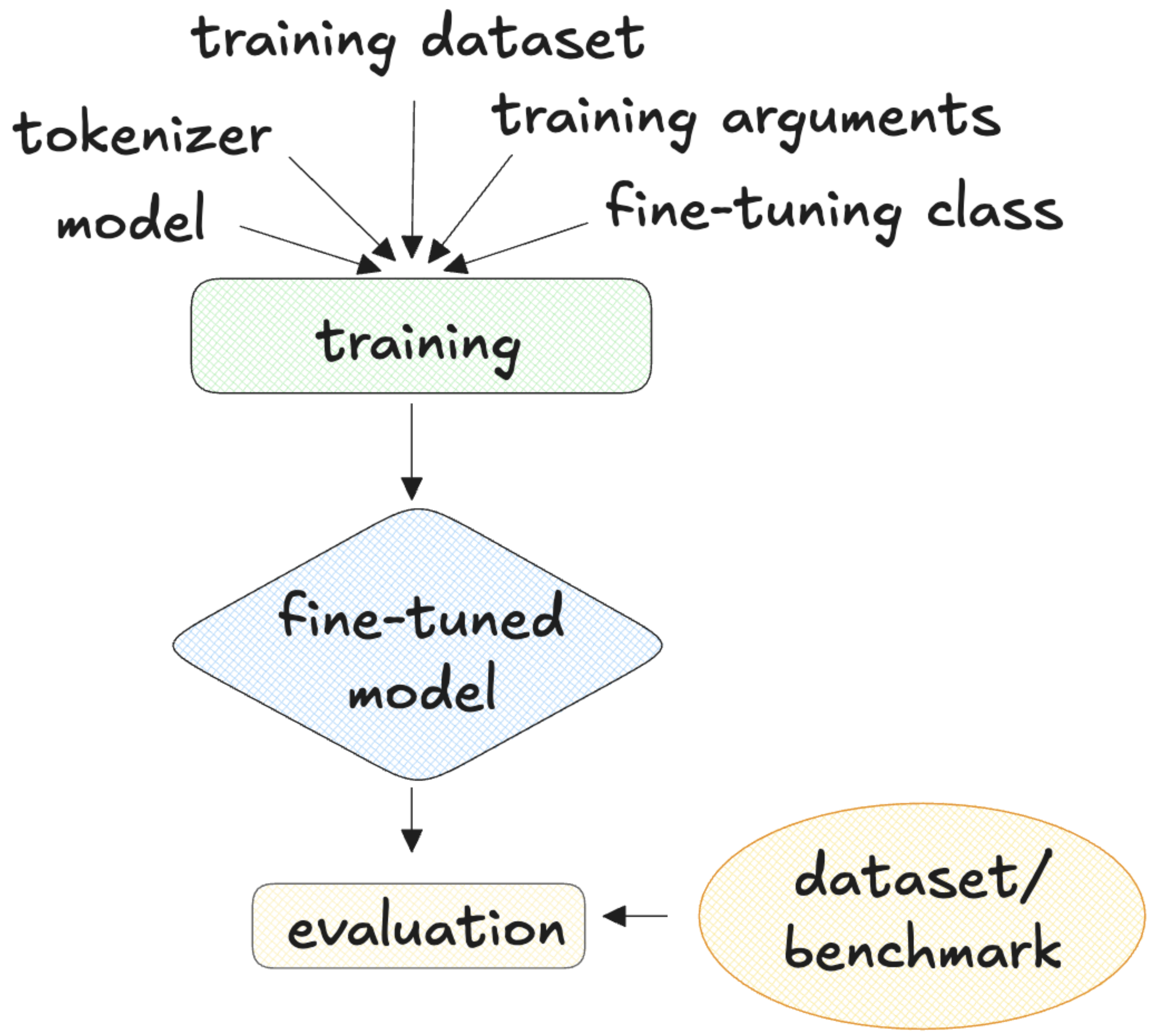
- Fine-tuning process



How to use fine-tuning

- Quality of the data
- Model's capacity
- Task definition

- Fine-tuning process
- New model
- Evaluation



The Llama fine-tuning libraries

- ▯ Several libraries for fine-tuning
- ▯ TorchTune for Llama fine-tuning
- ▯ Launching a fine-tuning task with TorchTune

Options for Llama fine-tuning

- **TorchTune**
  - Based on configurable templates
  - Ideal for: **scaling quickly**
- **SFTTrainer from Hugging Face**
  - Access to other LLMs
  - Ideal for: **fine-tuning multiple models**
- **Unsloth**
  - Efficient memory usage
  - Ideal for: **limited hardware**
- **Axolotl**
  - Modular approach
  - Ideal for: **no extensive reconfiguration**

TorchTune and the recipes for fine-tuning

- TorchTune recipes:
    - Modular templates
    - Configurable to be adapted to different projects
    - Keep code organized
    - Ensure reproducibility

TorchTune list

- Run from a terminal
- Environment with Python
- Install TorchTune

```
pip3 install torchtune
```

- List available recipes

```
tune ls
```

- ! if using IPython

```
!tune ls
```

TorchTune list

```
!tune ls
```

- Output:

```
RECIPE                                  CONFIG
full_finetune_single_device             llama3/8B_full_single_device
                                        llama3_1/8B_full_single_device
                                        llama3_2/1B_full_single_device
                                        llama3_2/3B_full_single_device
full_finetune_distributed               llama3/8B_full
                                        llama3_1/8B_full
                                        llama3_2/1B_full
                                        ...
```

TorchTune run

- Use recipe + `--config` + configuration
- Run fine-tuning

```
tune run full_finetune_single_device --config \
llama3_1/8B_lora_single_device
```

- Parameters `device=cpu` or `device=cuda`
- `epochs=<int>` (`<int>` is 0 or a positive integer)

Let's practice!

FINE-TUNING WITH LLAMA 3

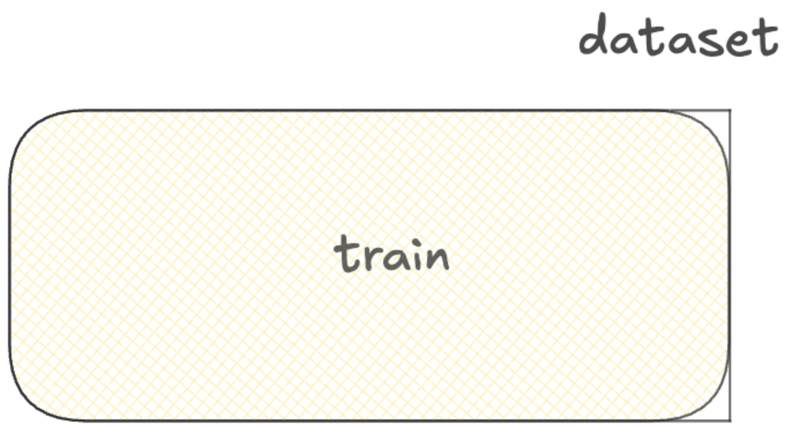
Preprocessing data for fine-tuning

FINE-TUNING WITH LLAMA 3

Francesca Donadoni
Curriculum Manager, DataCamp

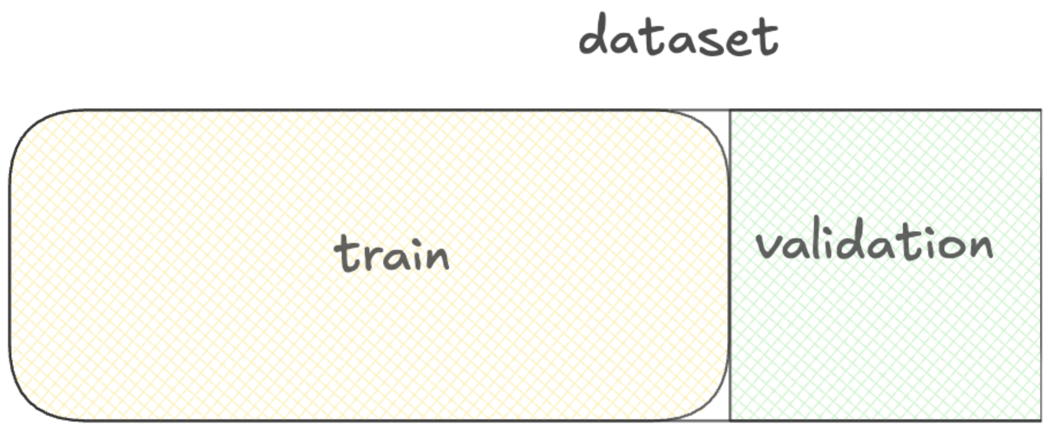
Using datasets for fine-tuning

- **Quality of the data** is key
- **Training Set:**
  - For model training
  - Majority of the data



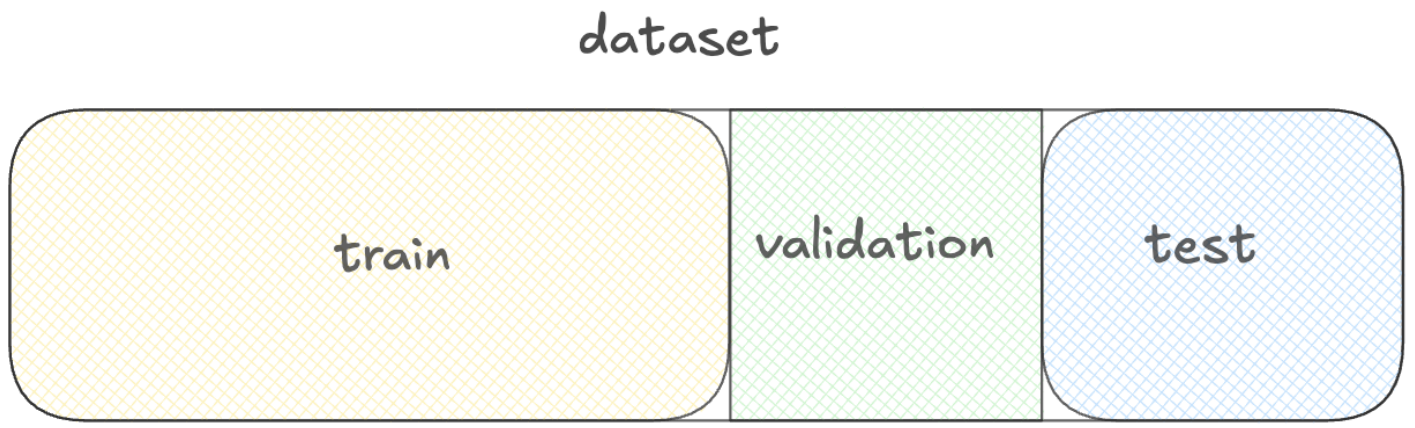
Using datasets for fine-tuning

- **Quality of the data** is key
- **Training Set:**
  - For model training
  - Majority of the data
- **Validation Set:**
  - For selecting the best model version



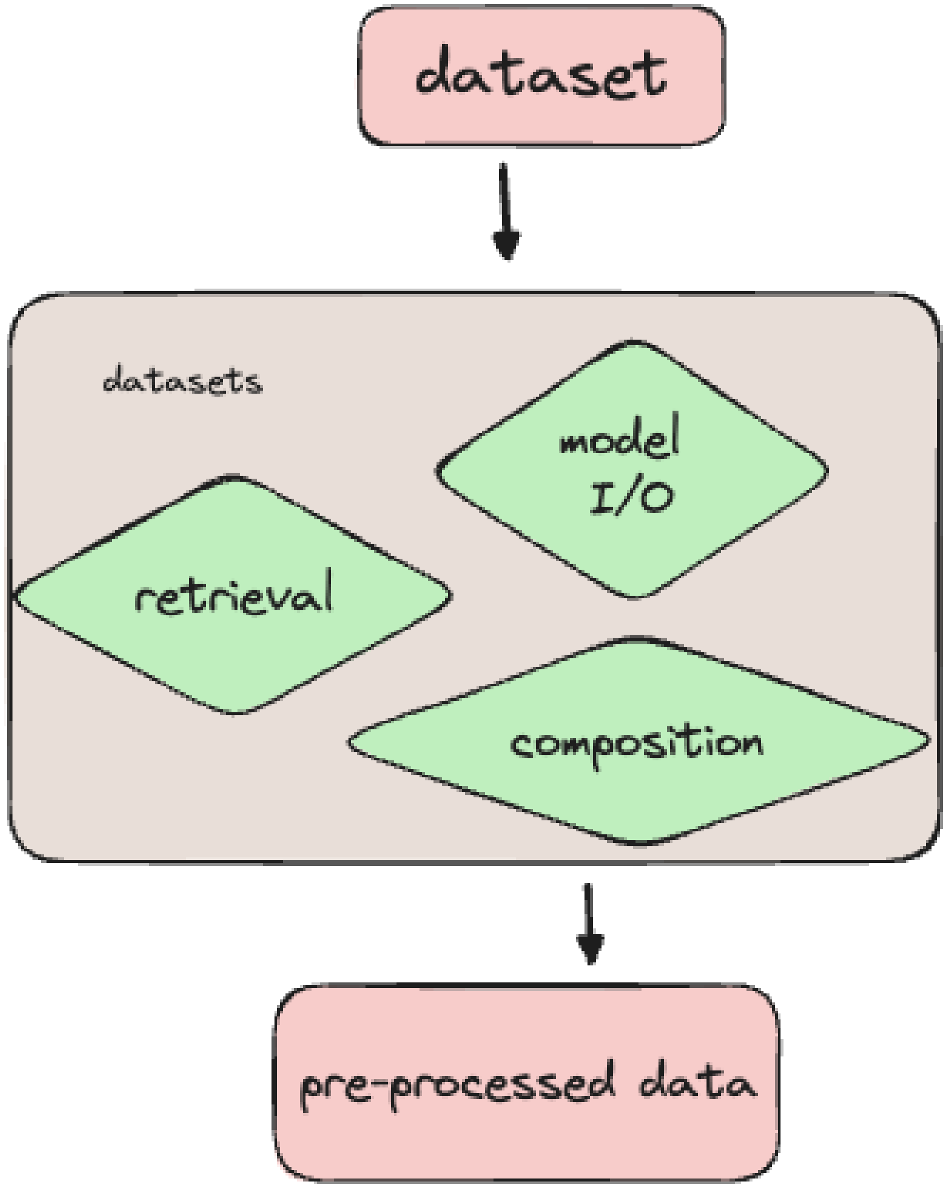
Using datasets for fine-tuning

- **Quality of the data** is key
- **Training Set:**
  - For model training
  - Majority of the data
- **Validation Set:**
  - For selecting the best model version
- **Test Set:**
  - For evaluating model's performance



Preparing data using the datasets library

- Datasets library
- Preprocessing
- Split
- Load
- Manage memory



Loading a customer service dataset

```python
from datasets import load_dataset
ds = load_dataset(
    'bitext/Bitext-customer-support-llm-chatbot-training-dataset',
    split="train"
)
print(ds.column_names)
```

```
['flags', 'instruction', 'category', 'intent', 'response']
```

Peeking into the data

```python
import pprint
pprint.pprint(ds[0])
```

```
{'category': 'ORDER',
 'flags': 'B',
 'instruction': 'question about cancelling order {{Order Number}}',
 'intent': 'cancel_order',
 'response': "I've understood you have a question regarding canceling order "
             "{{Order Number}}, and I'm here to provide you with the "
             'information you need. Please go ahead and ask your question, and '
             "I'll do my best to assist you."}
```

Filtering the dataset

```python
from datasets import load_dataset, Dataset

ds = load_dataset(
    'bitext/Bitext-customer-support-llm-chatbot-training-dataset',
    split="train")
print(ds.shape)
```

```
(26872, 5)
```

```python
first_thousand_points = ds[:1000]
ds = Dataset.from_dict(first_thousand_points)
```

Preprocessing the dataset

```python
def merge_example(row):
    row['conversation'] = f"Query: {row['instruction']}\nResponse: {row['response']}"
    return row
ds = ds.map(merge_example)
print(ds[0]['conversation'])
```

```
Query: question about cancelling order {{Order Number}}
Response: I've understood you have a question regarding canceling order {{Order Number}},
and I'm here to provide you with the information you need. Please go ahead and ask your
question, and I'll do my best to assist you.
```

Saving the preprocessed dataset

```
ds.save_to_disk("preprocessed_dataset")
```

```
Saving the dataset (1/1 shards): 100%
26872/26872 [00:00<00:00, 383823.33 examples/s]
```

```
from datasets import load_from_disk
ds_preprocessed = load_from_disk("preprocessed_dataset")
```

Using Hugging Face datasets with TorchTune

- Can use Hugging Face dataset with TorchTune
- Set a dataset path and configurations

```
tune run full_finetune_single_device --config llama3/8B_full_single_device \
dataset=preprocessed_dataset dataset.split=train
```

Let's practice!

FINE-TUNING WITH LLAMA 3

Fine-tuning with TorchTune

FINE-TUNING WITH LLAMA 3

Francesca Donadoni
Curriculum Manager, DataCamp

The components of TorchTune fine-tuning

- **Model**
  - Defines the architecture and pre-trained weights to fine-tune
  - Different versions and number of parameters available
- **Dataset**
  - Specifies the data used for training
- **Recipe**
  - Central configuration file combining the model, dataset, and training parameters
  - Ensures consistency and reproducibility

The components of TorchTune fine-tuning

- **Model**
  - `!tune ls`

```
llama3/8B_full
llama3_1/8B_full
llama3_2/1B_full ...
```

- **Dataset**
  - `ds.save_to_disk("new_dataset")`
- **Recipe**
  - `custom_recipe.yaml`

The components of a TorchTune recipe

- General Settings and ouput directory
  - Batch size, device and epochs
- Model
  - Specifies architecture and model configurations
- Optimizer
  - Includes learning rate
- Dataset
  - Defines preprocessing and dataset path

```
batch_size: 4
device: cuda
epochs: 20
output_dir: /tmp/full-llama3.2-finetune

model:
  _component_:
      torchtune.models.llama3_2.llama3_2_1b

optimizer:
  _component_: bitsandbytes.optim.PagedAdamW8bit
  lr: 2.0e-05

dataset:
  _component_: torchtune.datasets.alpaca_dataset
```

Configuring TorchTune recipes

- More parameters available
- Configurable in Python using `yaml`

```python
import yaml
config_dict = {"batch_size": 4,
               "device": "cuda",
               "model": {
                 "_component_": "torchtune.models.llama3_2.llama3_2_1b"
                },
                ...
}
yaml_file_path = "custom_recipe.yaml"
with open(yaml_file_path, "w") as yaml_file:
    yaml.dump(config_dict, yaml_file)
```

Running custom fine-tuning

```
tune run --config custom_recipe.yaml
```

```
INFO:torchtune.utils.logging:Running
Writing logs to /tmp/full-llama3.2-finetune/log_1732815689.txt
INFO:torchtune.utils.logging:Model is initialized with precision torch.bfloat16.
INFO:torchtune.utils.logging:Tokenizer is initialized from file.
1|52|Loss: 2.3697006702423096:   0%|?                     | 52/25880
```

- Saved logs
- Successful initialization
- Epoch and step count progress
- Loss metrics

Let's practice!

FINE-TUNING WITH LLAMA 3