LARGE LANGUAGE MODELS

FINETUNING & SERVING RECIPE

Quan Nguyen

ABOUT ME

Quan Nguyen

VILM, Zalo AI

- Product Executive @ Zalo AI
- Co-Founder @ Virtual Interactive

- Former AI Resident @ FPT Software AI Center
- Former Research Engineer @ OpenAI

AGENDA

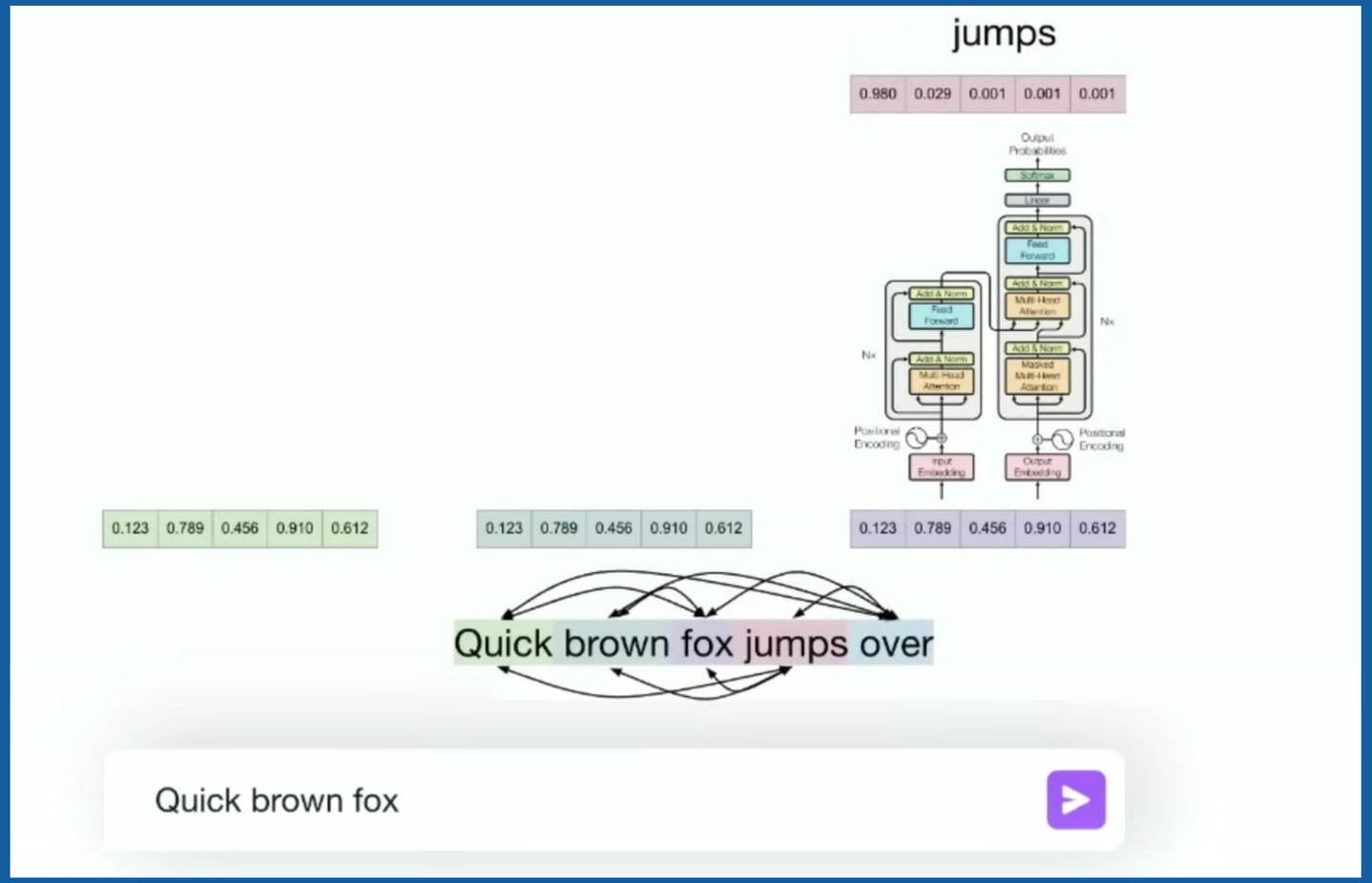
1) TRAINING

- DATA: DATA PREPARATION, DATA GENERATION
- LIBRARIES & COMPUTATION: FINETUNE, EMBEDDINGS & GPU ESTIMATION

2) SERVING

- RETRIEVAL AUGMENTED GENERATION
- SERVING LIBRARIES: TGI, TEI, VLLM & LLAMA.CPP

LARGE LANGUAGE MODELS (LLMS)

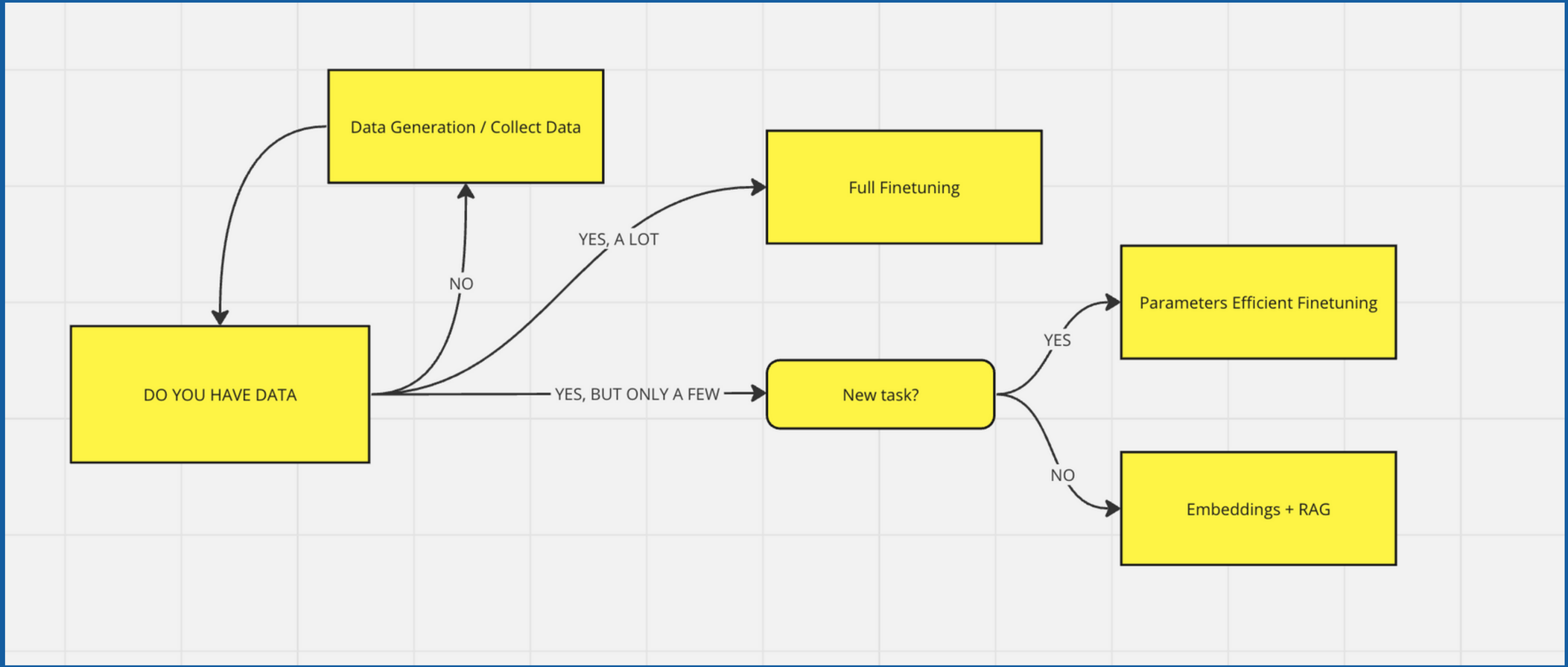


WHY FINETUNING LLMS?

- PRETRAINED MODELS ARE POWERFUL BUT:
  - GOOD GENERALIST, BUT BAD SPECIALIZATION
  - NEED GOOD KNOWLEDGE-BASE TO BE ABLE TO PERFORM WELL IN NEW TASKS

- FINETUNING:
  - CHEAP AND EFFICIENT
  - ADAPT PRETRAINED MODELS TO REAL WORLD APPLICATIONS
  - UNLEASH THE POTENTIAL OF LLMS

DESIGN A EFFICIENT FINETUNING PIPELINE



DATA GENERATION

- It is hard to find quality data in the wild, especially instructional ones. Often, those data would requires heavy cleaning process. This usually takes a lot of time
- Good data = Good performance
- Synthetic data allows us to control the constrain of the data, making it even safer than data in the wild

-> Data generation and distillation from larger models will make sure you have a good quality dataset while minimize your time spending on processing the data.

Example: Orca, Phi-1.5, Evol-Instruct

DATA GENERATION

- LLM prompts usally consists of 3 parts: System Instruction (Optional), Input (question), and Output (answer):
  - SYS (Optional): Make sure you define the characteristics of the chatbot/assistant. This is important if you want to build a generalist LLM
  - Input: The question that the user asks
  - Output: The answer that the assistant will response given the input

DATA PREPARATION

- Prepare datasets in common formats also will minimize the time and efforts in finetuning data since these formats are widely supported by the community
- There are two common formats in organizing LLM finetuning data:
  - ShareGPT - most common
  - Alpaca

DATA PREPARATION - SHAREGPT

```
[
  {
    "id": "identity_0",
    "conversations": [
      {
        "from": "human",
        "value": "Who are you?"
      },
      {
        "from": "gpt",
        "value": "I am Vicuna, a language model trained by researchers from Large Model Systems Organization (LMSYS)."
      },
      {
        "from": "human",
        "value": "Have a nice day!"
      },
      {
        "from": "gpt",
        "value": "You too!"
      }
    ]
  },
  {
    "id": "identity_1",
    "conversations": [
      {
        "from": "human",
        "value": "Who are you?"
      },
      {
        "from": "gpt",
        "value": "My name is Vicuna, and I'm a language model developed by Large Model Systems Organization (LMSYS)."
      }
    ]
  },
```

DATA PREPARATION - ALPACA

Search this dataset

instruction string · *lengths* 9↔58 51.7%	input string · *lengths* 0↔247 98.8%	output string · *lengths* 0↔419 74.5%	text string · *lengths* 154↔589 67.2%
Identify the odd one out.	Twitter, Instagram, Telegram	Telegram	Below is an instruction that describes a task,…
Explain why the following fraction is equivalent to 1/4	4/16	The fraction 4/16 is equivalent to 1/4 because both numerators and denominators are divisible by 4. Dividing both the top and bottom numbers by 4 yields the fraction 1/4.	Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. ### Instruction: Explain why the following fraction is equivalent to 1/4 ### Input: 4/16 ### Response: The fraction 4/16 is equivalent to 1/4 because both numerators and denominators are

LIBRARIES - PRETRAINING/FINETUNING

- **Axolotl** is the leading open-source library for training/finetuning LLM
- Offers a lot of optimization and pipelines to speed up your finetuning process: DeepSpeed, PEFT, FlashAttention 2, wandb, etc.
- Start training in just 3 minutes if you followed the correct data/model format

Axolotl supports

	fp16/fp32	lora	qlora	gptq	gptq w/flash attn	flash attn	xformers attn
llama	✅	✅	✅	✅	✅	✅	✅
Pythia	✅	✅	✅	❌	❌	❌	?
cerebras	✅	✅	✅	❌	❌	❌	?
btlm	✅	✅	✅	❌	❌	❌	?
mpt	✅	❌	?	❌	❌	❌	?
falcon	✅	✅	✅	❌	❌	❌	?
gpt-j	✅	✅	✅	❌	❌	?	?
XGen	✅	?	✅	?	?	?	✅
phi	✅	✅	✅	?	?	?	?

Dataset

Axolotl supports a variety of dataset formats. Below are some of the formats you can use. Have dataset(s) in one of the following format (JSONL recommended):

- `alpaca`: instruction; input(optional)

```
{"instruction": "...", "input": "...", "output": "..."}
```

- `sharegpt`: conversations where `from` is `human` / `gpt`

```
{"conversations": [{"from": "...", "value": "..."}]}
```

- `completion`: raw corpus

```
{"text": "..."}
```

LIBRARIES - REINFORMENT LEARNING

- For further enhancement in quality/safety of the answers, Reinforment Learning could be used

- HuggingFace's TRL is the leading library for RL for Transformers

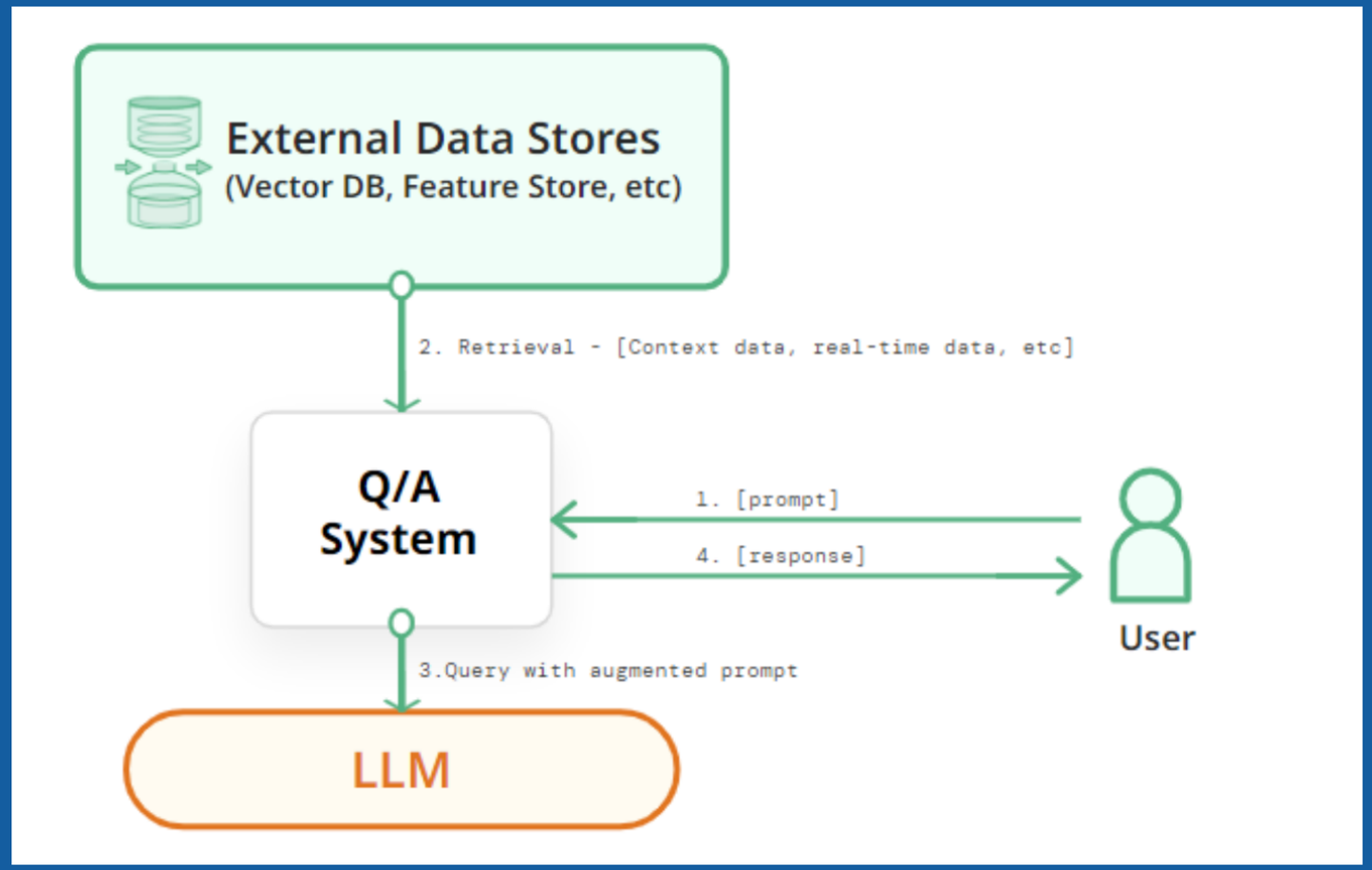
EMBEDDINGS

- Finetuning is not the only choice. Sometimes, finetuning is not even the best solution

- Rule of Thumb: If you do not have a lot of data (>1M samples), and your task is QA on a specific source (PDF, text, or anything can be turned into text) -> RAG Embedding is the best choice

RETRIEVAL AUGMENTED GENERATION (RAG)

- RAG is the processing of adding more knowledge to the model's knowledge-base by leveraging search indexing and in-context learning



From Simple to Advanced

Table Stakes

Prompt Engineering

Better Parsers
Chunk Sizes
Hybrid Search
Metadata Filters

Advanced Retrieval

Reranking
Recursive Retrieval
Embedded Tables
Small-to-big Retrieval

Fine-tuning

Embedding fine-tuning
LLM fine-tuning

Agentic Behavior

Routing
Query Planning
Multi-document Agents

Less Expressive
Easier to Implement
Lower Latency/Cost

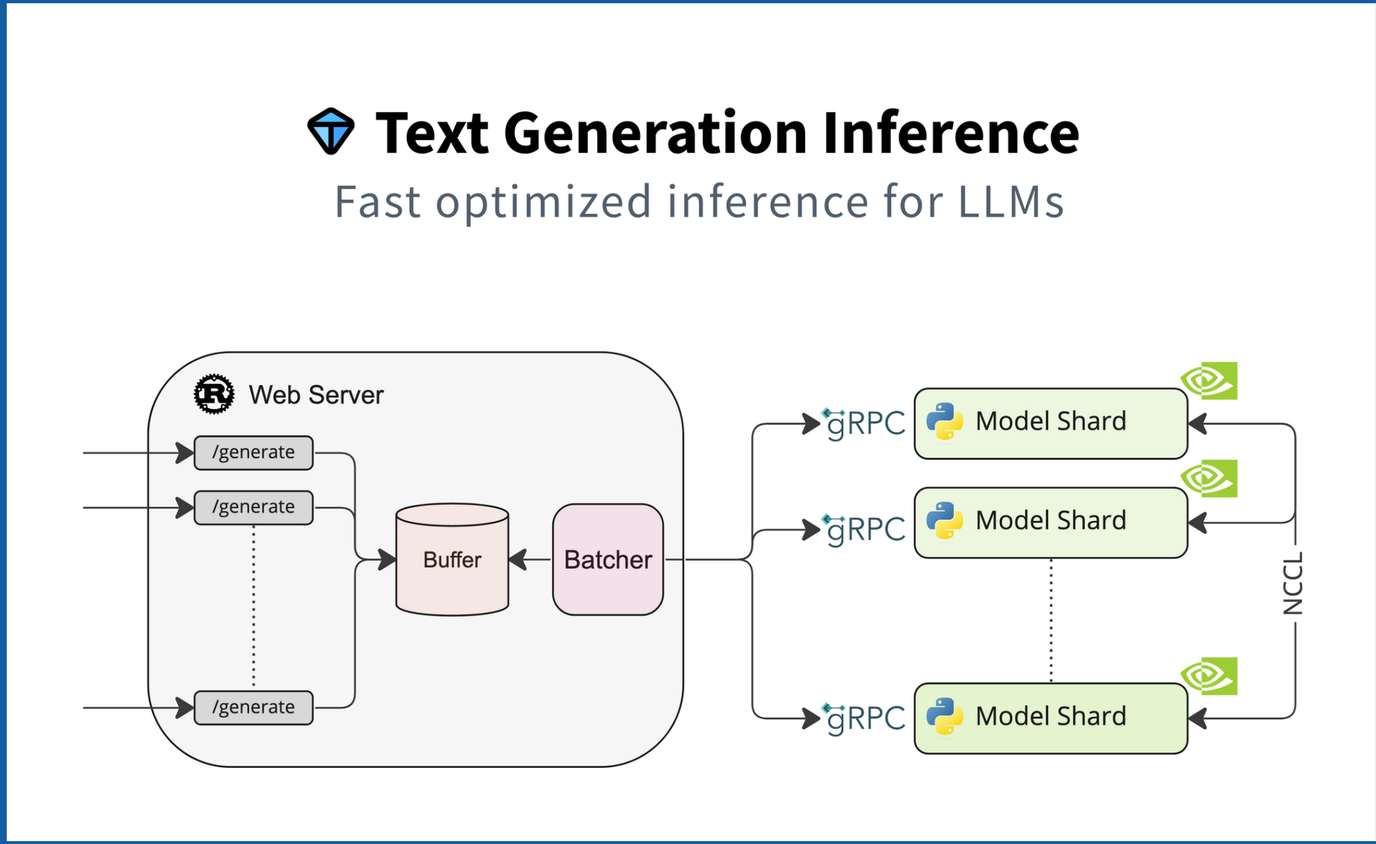
More Expressive
Harder to Implement
Higher Latency/Cost

SERVING

- It’s the best if your model is in the HuggingFace’s format. If not, DO IMMIDIATELY. Why?
- HuggingFace is the largest open-source AI organization in the world. Everything is optimized for HuggingFace
- Serving with HF’s models are **extremely easy**

SERVING - TEXT GENERATION INFERENCE (TGI)

- TGI is the serving backend for HF's compatible models.
- Setting up Inference Backend with TGI takes only 5 minutes



```
model=tiiuae/falcon-7b-instruct
volume=$PWD/data # share a volume with the Docker container to avoid downloading weights every run

docker run --gpus all --shm-size 1g -p 8080:80 -v $volume:/data ghcr.io/huggingface/text-generation-
```

SERVING - LLAMA.CPP

- If GPU is the constrain. You might want to use C++ for your inference job
- **LLaMA.cpp** is an excellent backend if you need fast CPU inference.
- Also support quantization of models for even better performance

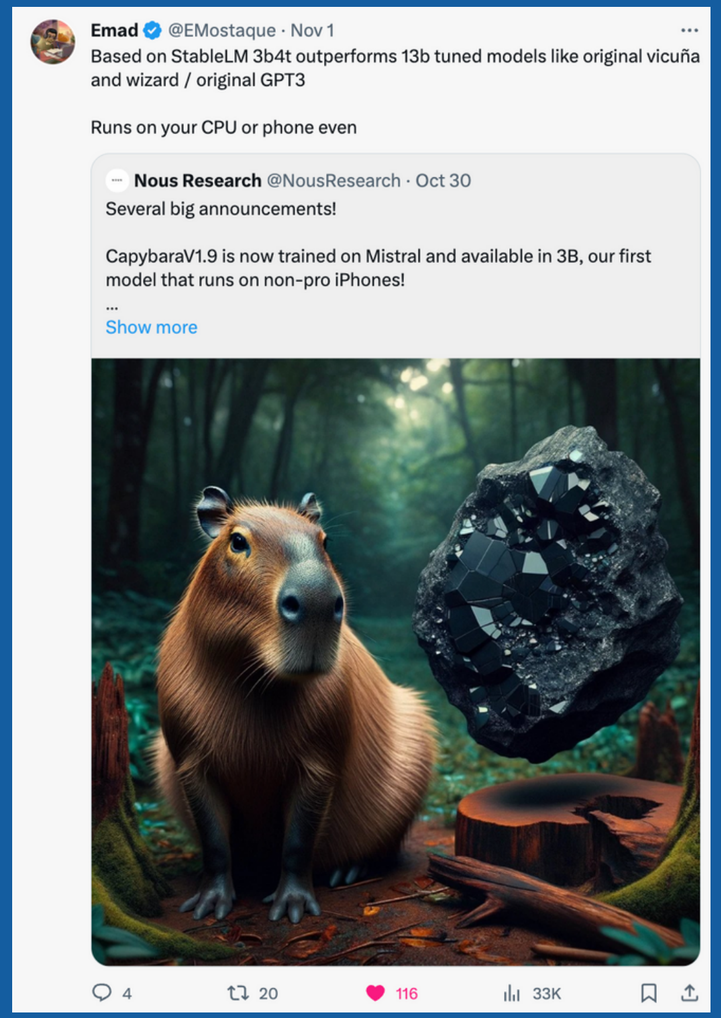
Drawbacks:

- Models need to be supported manually
- Quantization might decrease the quality of the answers

PUBLISH YOUR FIRST MODEL

- Publishing your first model might seem like an easy task, but will require a lot of time in preparation, make sure you have:
    - **Model Card on HuggingFace**
    - Benchmarks (if available)
    - **Inference Code (GitHub)**
    - Papers (if aplicable)

IF YOU FOLLOW THE PROCESS, HERE IS THE RESULT



THANK YOU FOR LISTENING

QnA Time