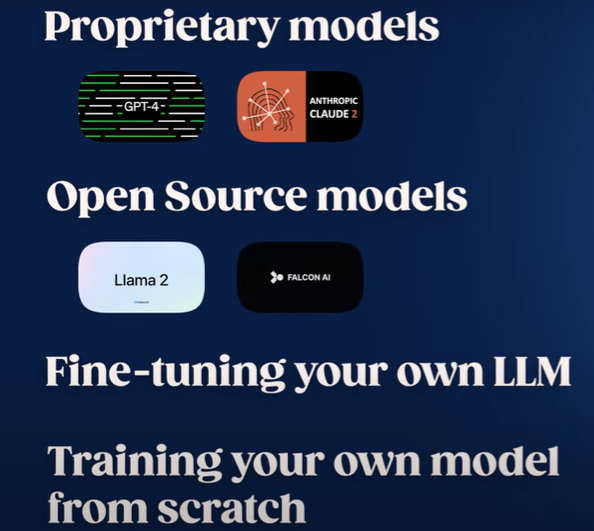
<https://youtu.be/Nr3ckDhDfK8>



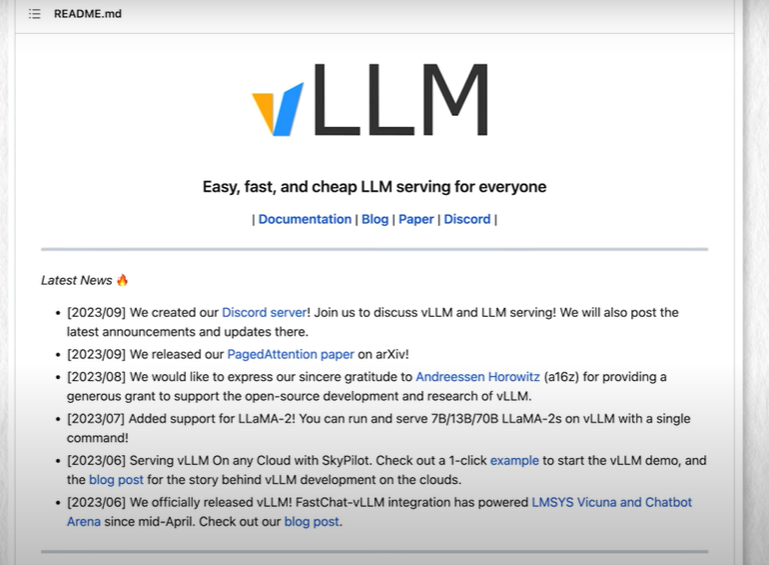
**Model Selection phase:**

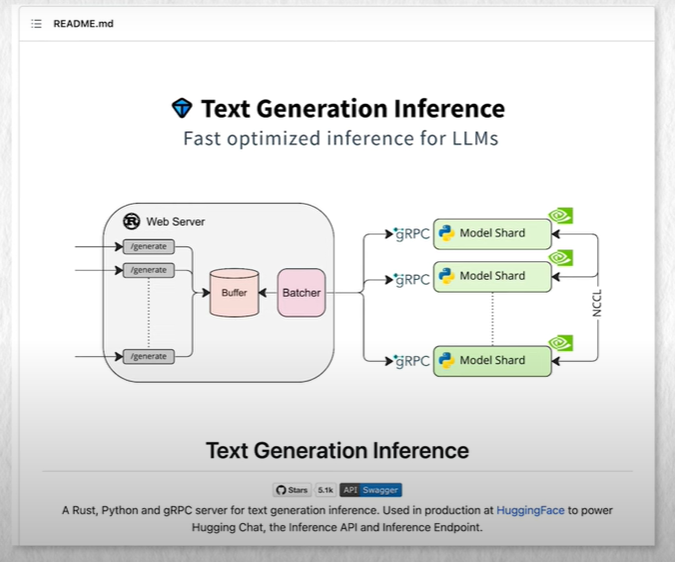
****

**Adaptation Phase:**

* **Finetuning**
* **RLHF**
* **RAG**
* **Deep Memory**

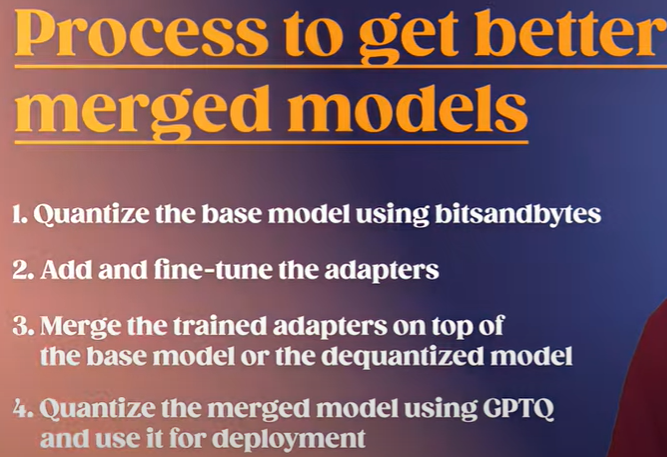


****

****

****

** **

****

****

Introduction to LLMOps

have you ever dreamed of launching a

company built around llms with those

powerful models now easily accessible

they are pretty much the golden ticket

to easily starting your own project in

this video we'll dive into all the steps

required to build your application based

on llms these steps Encompass the

practices commonly used to leverage llms

in production effectively and they are

typically referred to as large language

models operations or llm UPS from the

chat but powered by open eyes chity to

the smart writing assistant you love

like grammarly llms are reshaping almost

all Industries but creating a successful

llm based application is not that easy

it brings unique challenges that differ

from traditional products and even from

other AI based products here are the

five essential steps to Kickstart your

llm Venture which you need to understand

and carefully tackle to successfully

Implement by the way we've built an

entirely free course on training Ing

fine-tuning and deploying llms in

collaboration with tzi active Loop and

the Intel disruptor initiative linked

Below in which you can find Cod and

practical examples for all the steps I'm

discussing in this

video first you need to select the right

Step 1: Model selection phase.

llm for your use case here you have many

different choices from proprietary

models like gp4 by open AI or Cloud by

anthropic open source pre-trained llms

like latu or falcon fine-tuning your own

llm to even training your own model from

scratch which we all cover in our free

llm course train from scratch is very

difficult but it can definitely be a

GameChanger if you have the resource to

do it developers or startups often lean

towards proprietary models from Tech

Giants or open source Alternatives based

on platforms like hugging face

proprietary llms backed by substantial

Investments typically outperforms open

source versions and come with the added

benefit of cost-saving from not needing

to establish expensive inference

infrastructure and from economy of scale

additionally always check an lm's

knowledge cut off which is the last date

it was updated for instance chat GPT

can't discuss event past September 20121

which might lead to inaccurate outputs

on newer topics so it really depends on

your goal but you have many choices

hugging phase manages an online

leaderboard of open-source llms

evaluated on different curated benchmark

marks you may be interested in checking

it out to be always updated on the

latest open

Step 2: Adaptation phase.

llms once you choose a good foundation

model you must tailor it to your use

case once again you have different

options depending on the task I did a

full video to help you solve this exact

problem and better understand which

adaptation technique to use for your

task from F tuning prompting retraining

using reinforcement learning techniques

like rhf or reinforce ment learning from

AI feedback RL aif to using retrieval

augmented generation rag or its more

efficient alternative called Deep memory

from active Loop that we all explain in

detail in our free course to quickly

recap you can use fine tuning when you

want to make your llm an exper on a

specific topic you will want to use deep

memory when you have documentation for

your task that you want the model to use

and not hallucinate answers it's also

much cheaper than fine-tuning and can be

complimentary to it retraining from

scratch is already done but possible if

you want to entirely own your llm and

not rely on other companies and

approaches similar to what Bloomberg did

with Bloomberg GPD rhf and rla aif are

the powerful ways of fine tuning your

model to your task as I said we covered

these approaches in depth in the other

videos of the llm series if you want

more details in selecting the best

approach in your

Step 3: Evaluation phase.

case once your model is ready ready you

need to know how well it performs like

in school you need to compare it with

others using exams in this case the

exams are called benchmarks and just

like a philosophy exam rating the

students is super challenging since the

outputs are text answers which are

mainly subjective you cannot simply

classify the answer and voila it is

right or wrong for example try thinking

about how you could evaluate the quality

of an answer given by an llm assistant

whose job is to summarize YouTube videos

for which you don't have reference

summaries written by humans this is even

harder if your llm is supposed to work

as a general assistant like chat GPT

currently organizations often resort to

AB testing to assess the effectiveness

of their model checking whether the user

satisfaction is the same or better after

the change in production so you

minimally need to use multiple metrics

not just one to have a better overall

idea of the performance of your model

you also surely need qualitative

evaluations which means just play with

it and push it to its limit yourself as

I said you need to test your model on

multiple benchmarks that are related to

the task you want to tackle and compare

the metrics given to other approaches to

be sure you are somewhat competitive and

using the best possible solution at

least the best affordable solution here

again I have a complete video on the

different evaluation benchmarks for llms

and we have super practical examples for

doing that in the course

[Music]

Step 4: Deployment phase.

you now have your powerful model that

beats all others but it does that only

on your computer or remote server the

next step is to share it with the world

and this is called the deployment phase

which comes with lots of challenges from

latency to memory to cuss issues where

you need to make a lot of important

decisions deploying large language

models like GPT variant or any other llm

into real world applications often

requires a multi-stage process

you will integrate it into systems using

cloud-based apis such as Google vertex

AI or Amazon sagemaker or by deploying

the model directly using Frameworks like

tensorflow serving or Onyx all the

specific details will be dependent on

the size of your model and the speed of

responses you are looking for here are a

few challenges to look out for and tips

we gathered for you first compute

resources llms demand high computational

power ensure you have the necessary

infrastructure whether it's cloud-based

Solutions with AWS or Google cloud or

powerful local servers in practice for

smaller llms a standard GPU can be find

indeed an llm with 1 billion parameters

where each parameter is stored as a

float 32 requires 1 billion \* 4 BYT

which is 4 GB of memory for inference

which is fine for lower-end gpus

moreover by leveraging quantization

techniques it's possible to store the

model parameter ERS with smaller data

types like one bytes or 4 bits with

small downgrades in performance thus

saving even more in memory for example

using 4bit quantization we'd be able to

use an 8B parameter model on a GPU with

4 GB of R if you're looking at libraries

that can help you manage and deploy llms

you have the choice of vlm made by a

team of researchers and there's also the

text generation inference library from

the team at hugging face the shear size

of llms can make them slow and expensive

to run model destillation quantization

pruning or using smaller variants can

help you mitigate this which you can

learn more about in the course model

quantization is the simplest option you

can apply in order to reduce your

infrastructure costs and speed up the

inference when using open source llms

right now the two popular

implementations are bits and bytes and

gptq the team at hugging face published

a great article comparing the two

methods if you're interested they

conclude that bits and bytes is better

suited for fine-tuning while gptq is

better for Generation from their

observations one way to get better

merged models would be to first quantize

the model using bits and bytes add and

fine-tune the adapters merge the trained

adapters on top of the base model or

quantize the merged model using gptq and

use it for deployment then probably the

most important but underlooked challenge

ethical considerations llms can

sometimes produce biased or

inappropriate outputs continuous

monitoring and establishing ethical

guidelines are crucial you can also use

retrieval augmented approaches to help

mitigate hallucination and bias Problems

by the way I just published a video with

seven tips to help you mitigate that

source of llm Errors if you want to

learn more about that another important

Data privacy!

aspect to consider is data privacy when

fine tuning or doing continuous learning

on specific data ensure that user data

privacy is maintained and that you are

compliant with regulations like gdpr

speaking of continuous learning while

llms have fast knowledge they don't

learn from new data after deployment

unless retrained implementing a

continuous learning process can help

keep the model updated and increasingly

powerful you won't have the usual as of

my last update in September 2021 I do

not have realtime data about events or

elections that occurred after that point

message anymore if you you deployed your

model and checked for all these sources

of problems congrats the model is now

live and running but your work isn't

done

here you still need to monitor how your

Step 5: Monitoring phase.

model is performing online with new user

requests you will have bugs and

unexpected behavior that is for sure so

you need systems in place to visualize

and inspect the execution flow of your

LM analyze the inputs and outputs view

intermediate results and securely manage

prompts and and llm chain configurations

thankfully there are amazing companies

helping you do that and one that I

personally use is weights and biases and

more specifically weights and biases

prompts which offers a set of features

for developers to do all that you can

use any software you want but make sure

to track the llm and not let it be out

there it could scale up pretty quickly

and hurt lots of people again if you

want more information on that check out

the llm apps section of our course or

wait and biases directly mastering llm

Conclusion.

UPS is necessary for navigating the llm

based business landscape we've quickly

covered all the steps required to build

deploy and refine applications powered

by these AI Jugger notes but the

landscape is evolving quickly and

continuously so you must equip yourself

with the right tools and stay up to date

if this piqued your interest and you are

hungry for handson insights dive deeper

with our comprehensive course in

collaboration with 2zi active Loop and

the Intel disruptor initiative I hope

you've enjoyed this video of our llm

series stay tuned for more llm insights

in my upcoming

[Music]

[Music]

videos

- Generated with https://kome.ai