Generative Ai

@ nlpwith indrajit

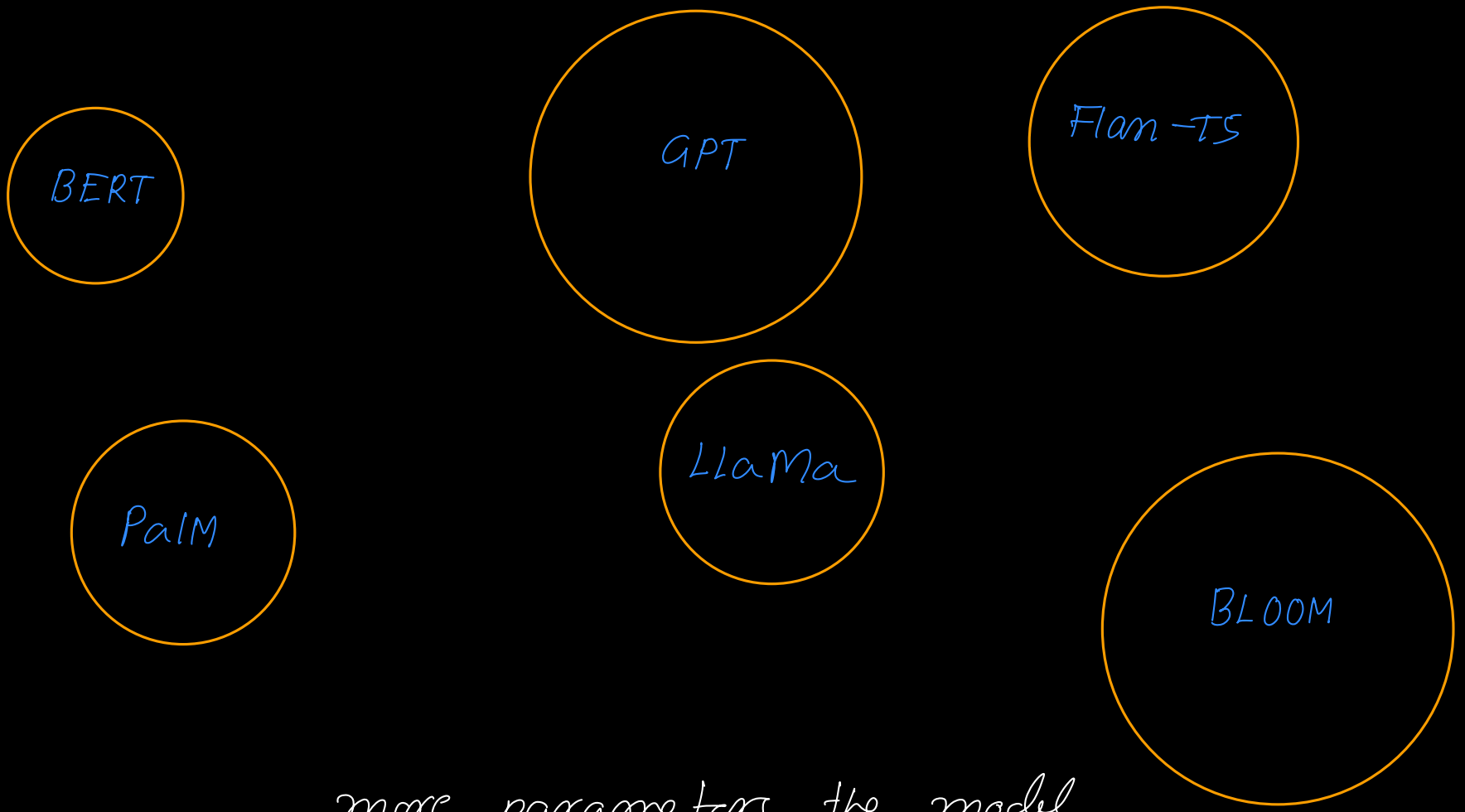
* Introduction ✓ Transformer / attention / looking at the important aspects of all this models.

* ✓ we will look into vision transformers

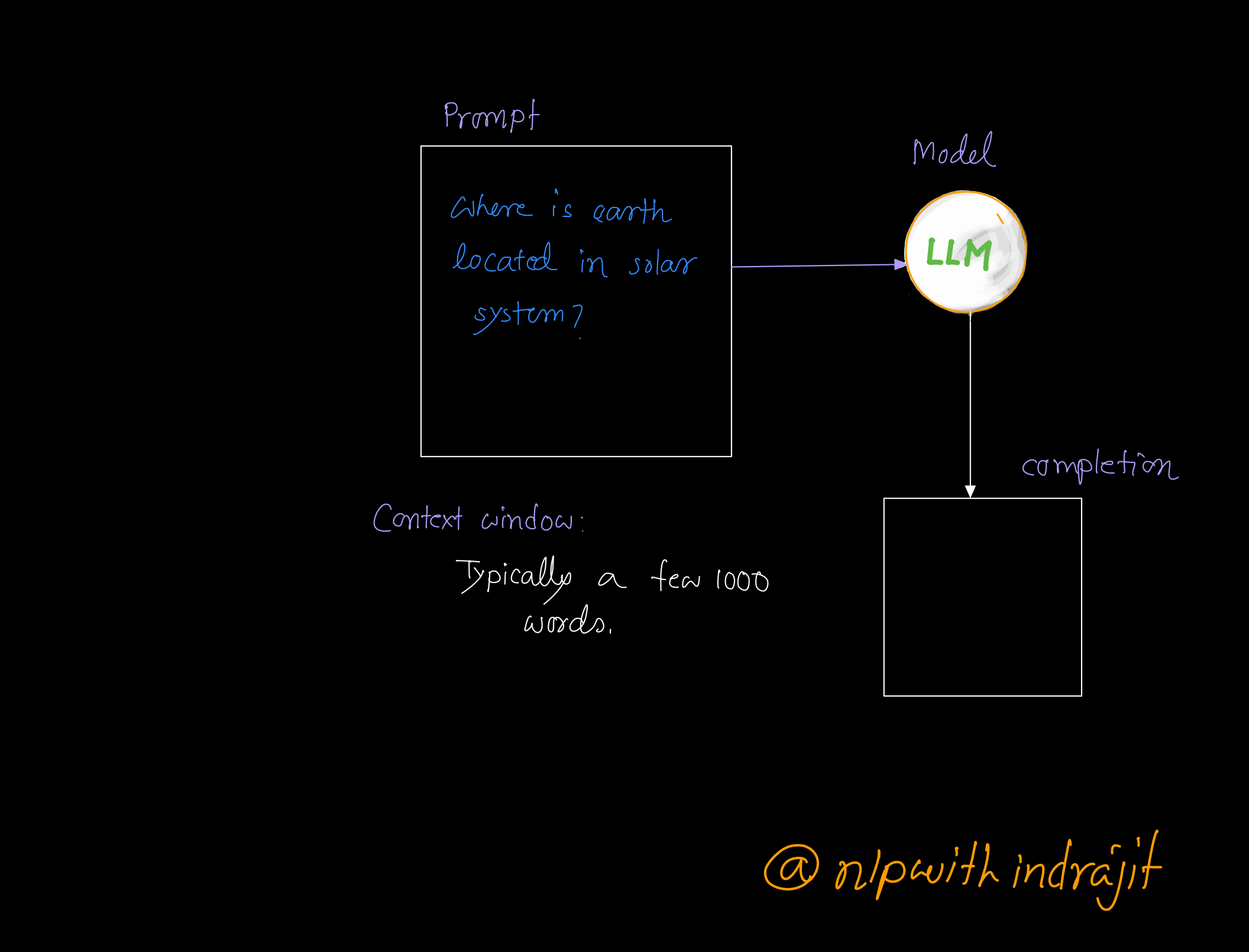
* ✓ Generative AI project life cycle
  - ✓ individual steps and
  - ✓ decisions that needs to be taken

@ nlpwith indrajit

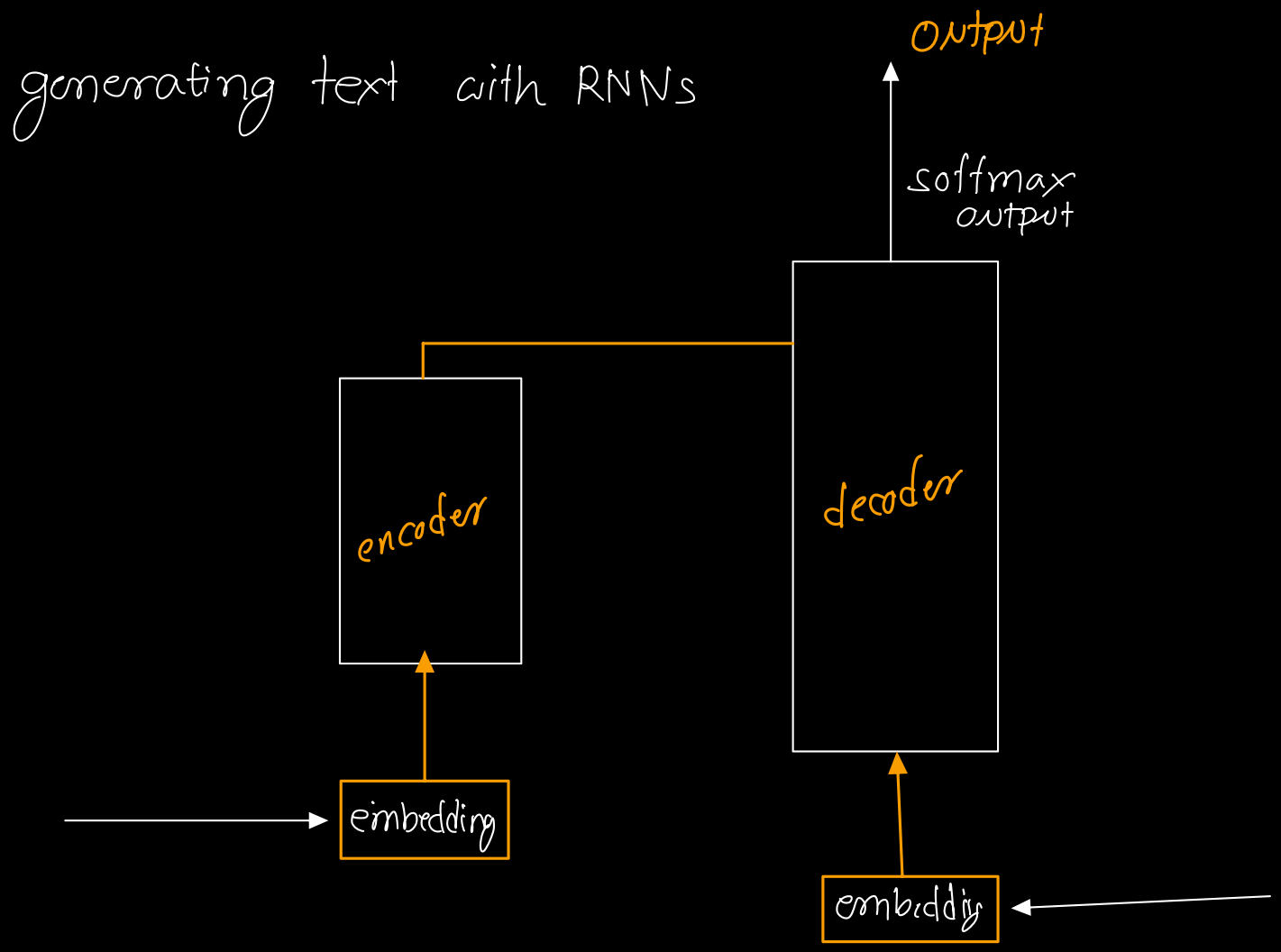
Large language models

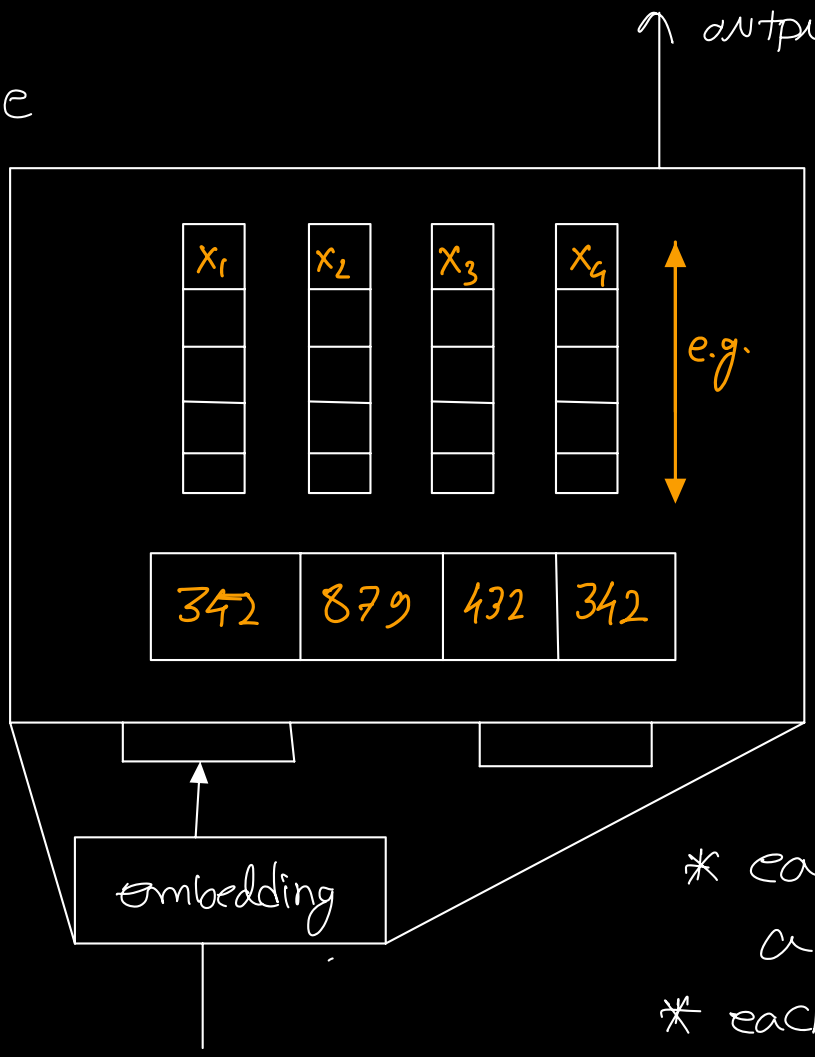


more parameters the model has, the more shophisticated work a model can do.


Prompt
Where is earth located in solar system?
Model
LLM
completion
Context window:
Typically a few 1000 words.
@ nlpwith indrajit

Text generation before transformers





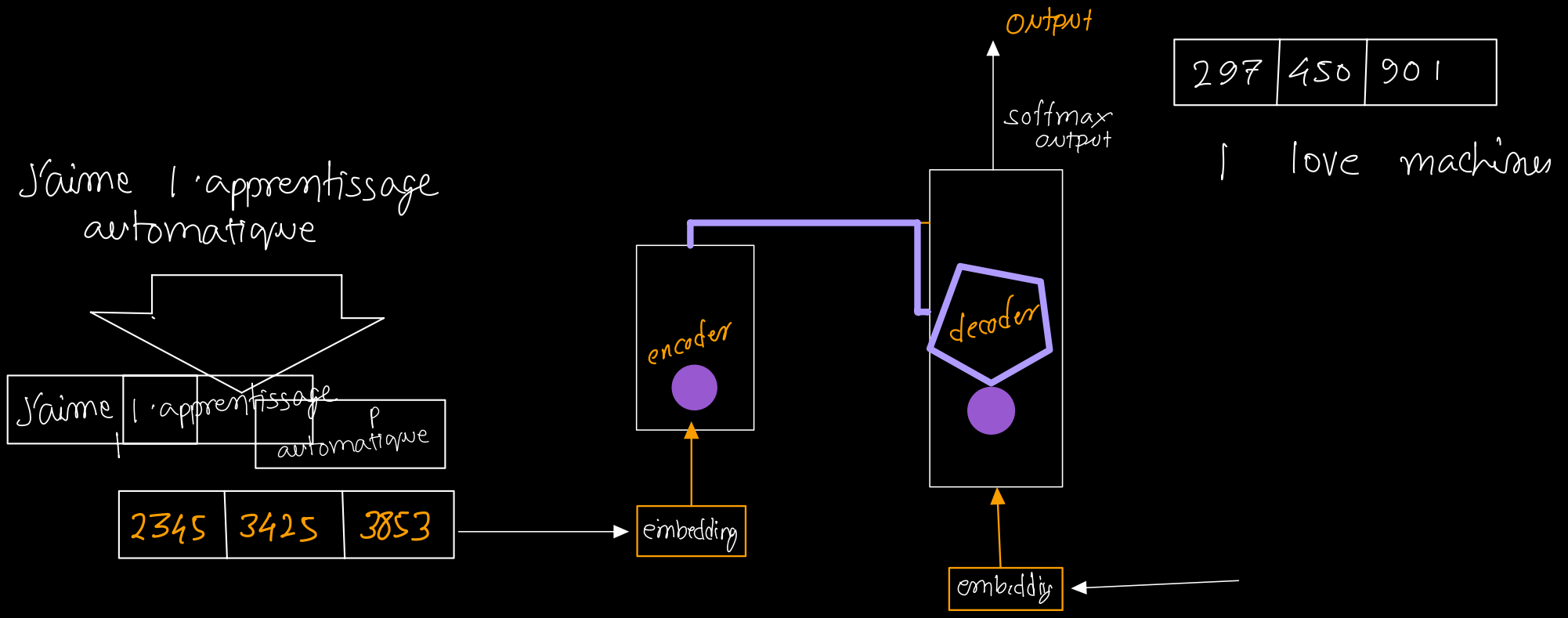
vector size (original tranformer paper it was 512)

* each word is mapped to a token ID
* each token is mapped into a vector.

positional embeddings helps in maintaining the word order.

Transformers

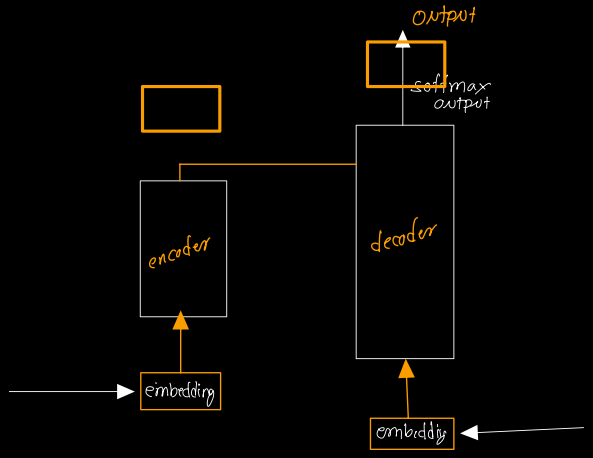
translate sequence to sequence tasks



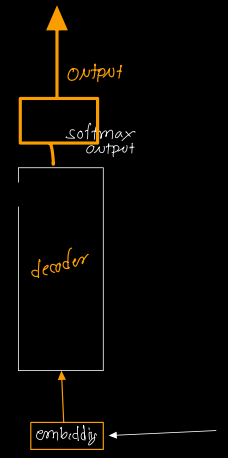
encoder only models works as a sequence to sequence (input sequence = output sequence)

BERT is an example of encoder only model ?

encoder only model



encoder decoder model



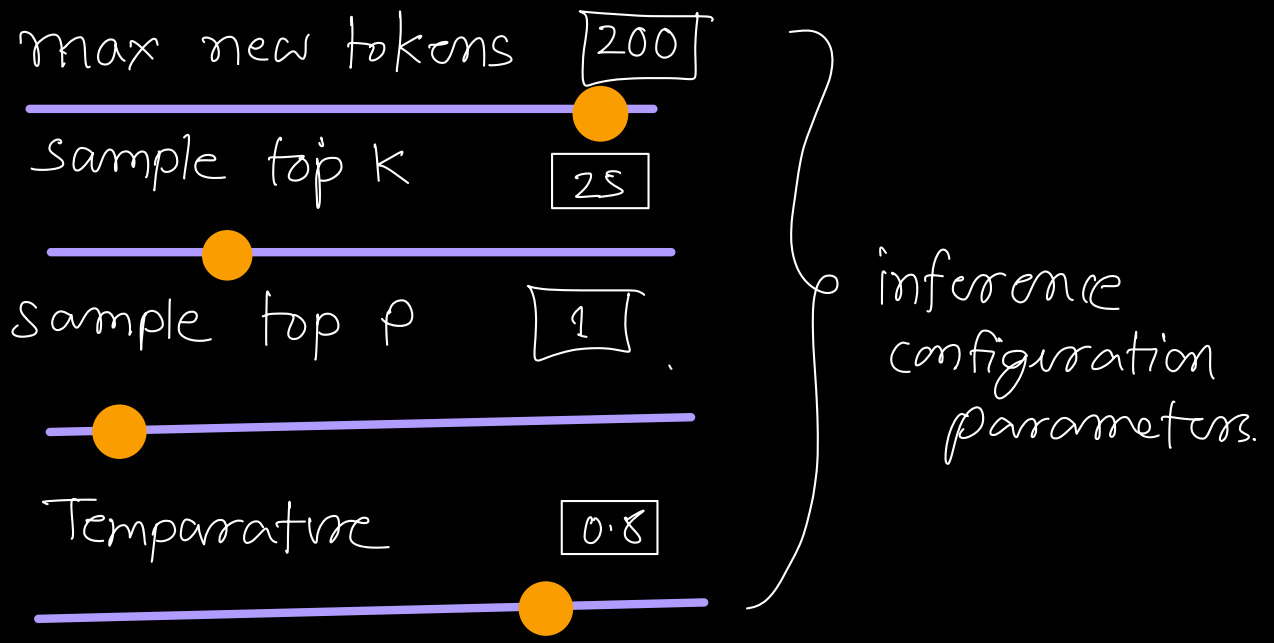
decoder model

In-context learning (ICL) → zero shot learning

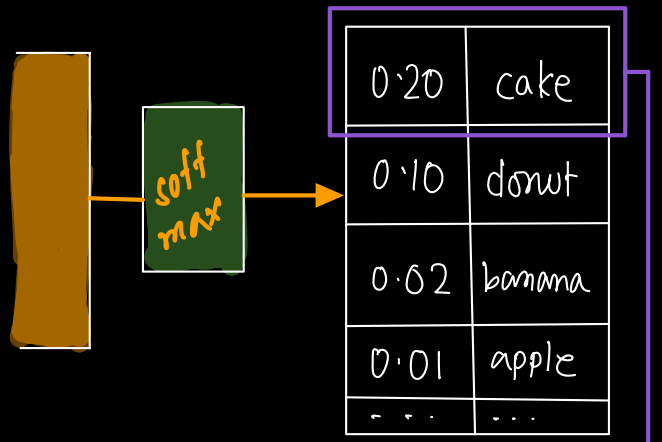
→ feashot learning

→ one shot learning

Generative Ai configuration – inference parameters



Set max new tokens

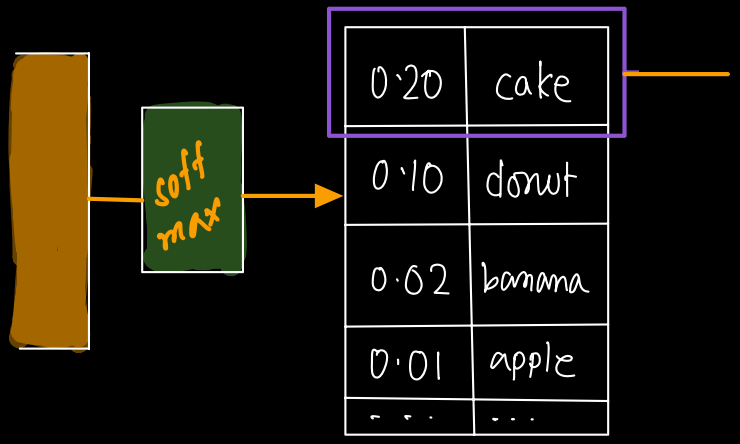


greedy: The word or token with highest probablity is selected.

most large language models will offer it with **greedy** approch.

↳ bad because repeated words

random sampling



random (-weighted) sampling

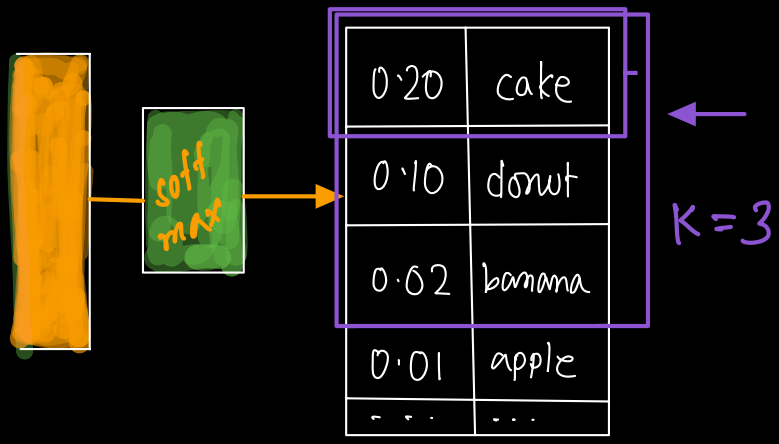
select a token using random -weighted strategy across the probabilities of all tokens.

Here there are 20% chance that cake will selected but 'banana' was actually selected

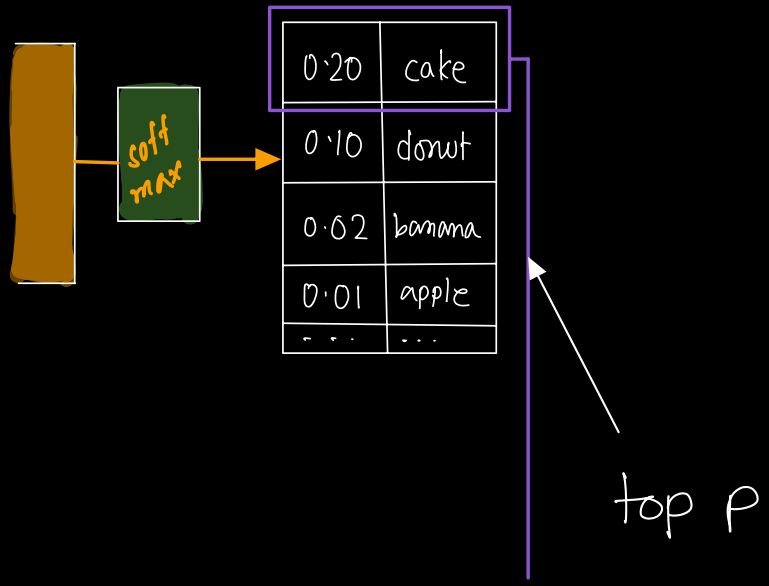
@ nlpwith indrajit

Sample top p

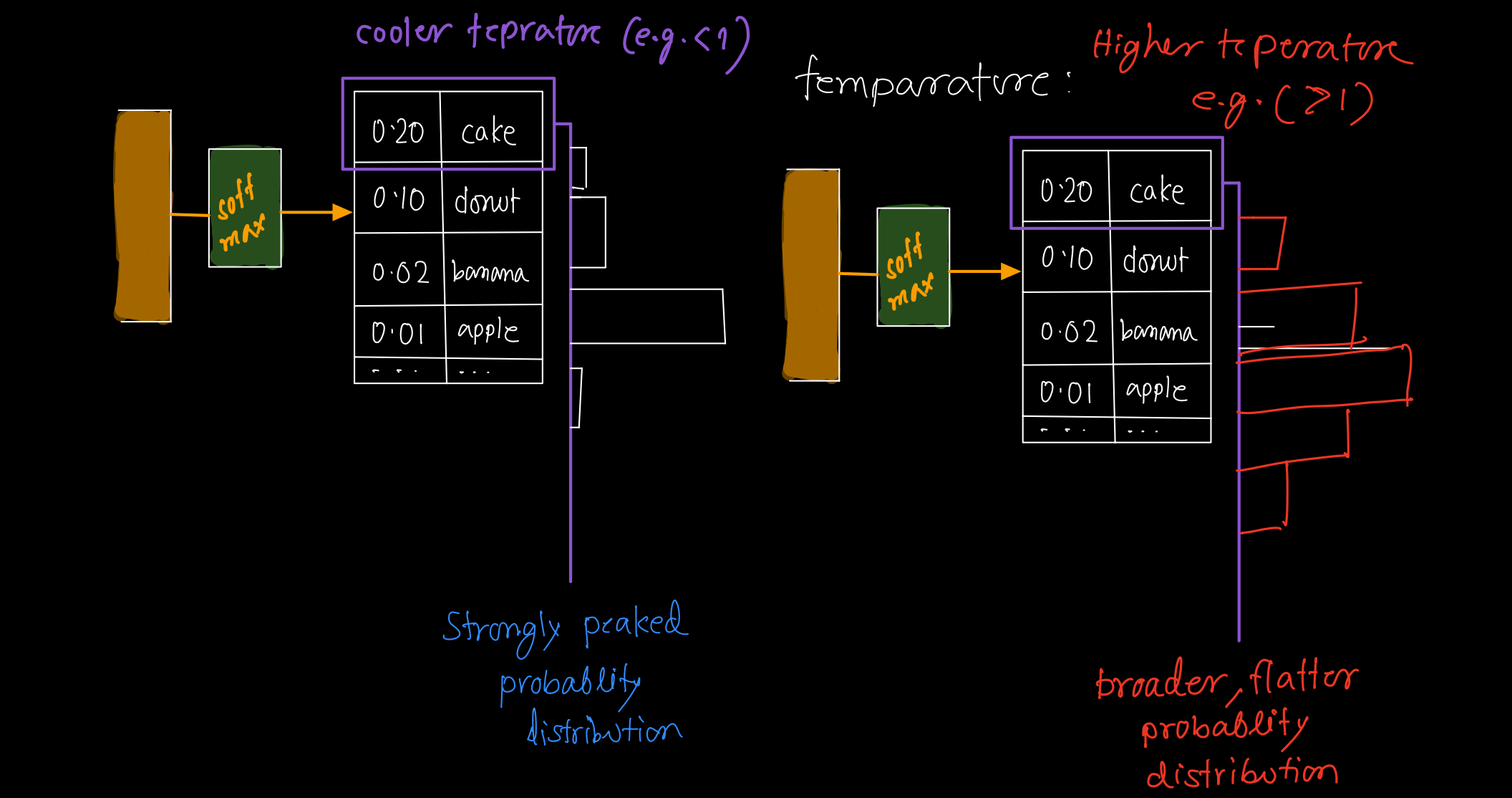
sample top k



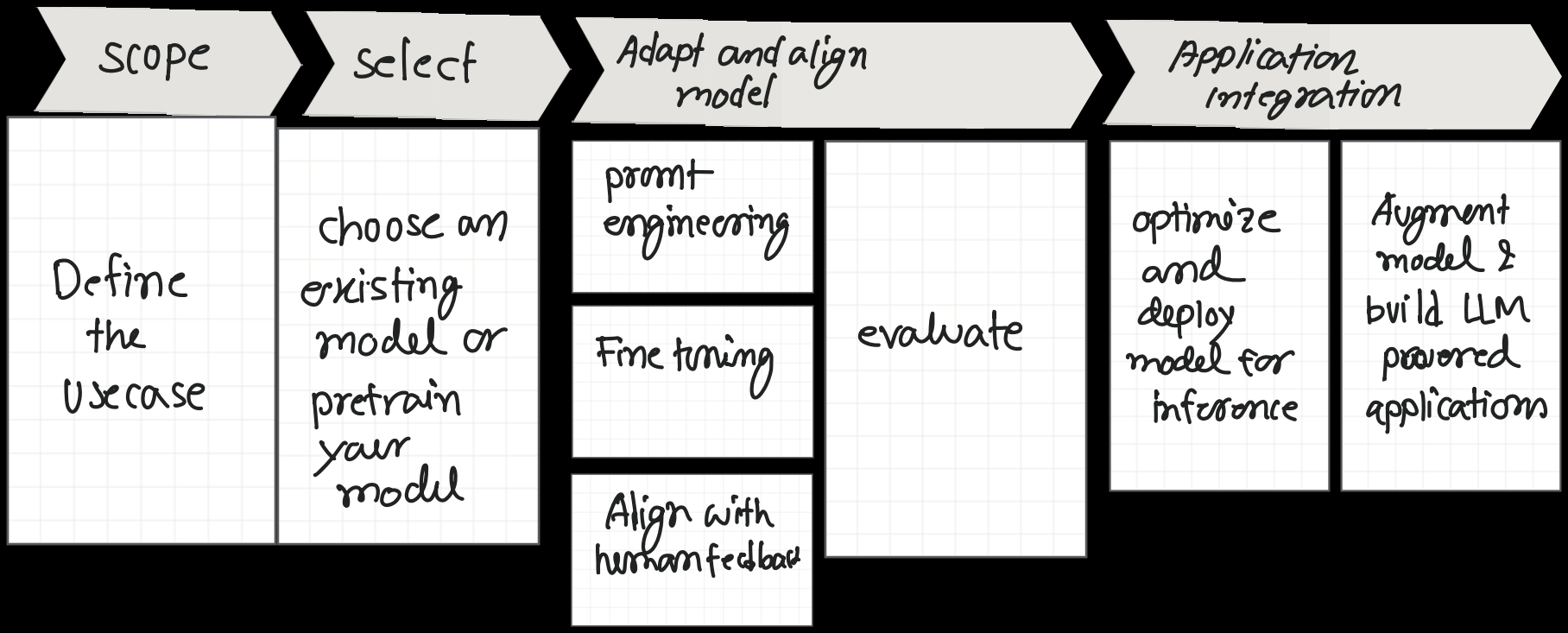
top-k: select an output from top-k results after applying random-weighted strategy using the probabilities.



top p: select an output using the random weighted strategy with the top-ranked consecutive results by probability and with a cumulative probability <= p


cooler temprature (e.g. <1)
0·20 cake
0·10 donut
0·02 banana
0·01 apple
soft max
Strongly peaked probability distribution
temparature:
Higher temperature e.g. (>1)
0·20 cake
0·10 donut
0·02 banana
0·01 apple
soft max
broader, flatter probability distribution

Generative AI project lifecycle



LLM Pretraining

Token String	Token ID	Embedding / vector representation
'_The'	37	[-0.0513, -0.0584, 0.0203, ...]
'_teacher'	3145	[-0.0513, -0.2344, 0.0313, ...]
'_teacher'	11749	[-0.0513, -0.12354, 0.0333, ...]
'_the'	8	[-0.433, -0.0584, 0.1233, ...]

Vocabulary

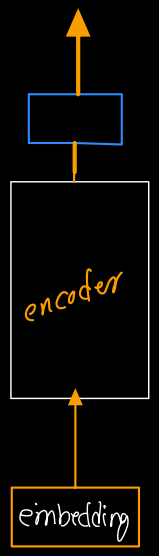
encoder only – autoencoders

encoder only models are also known as autoencoders

example: BERT / RoBERTA

Good use cases:

- Sentiment Analysis
- NER
- Word Classification



~> pretrained using MLM (Masked Language Modelling)

The	teacher	teaches	the	student

mask

Encoder only

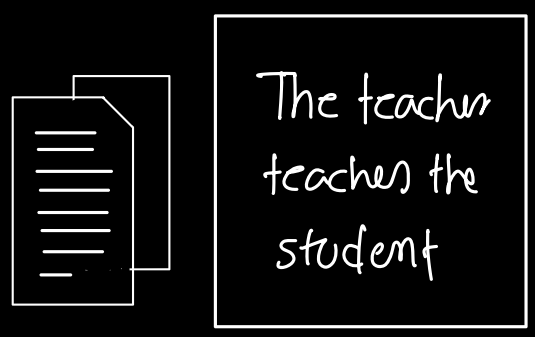
objective: Reconstruct text ("denoising")

The	teacher	teaches	the	student

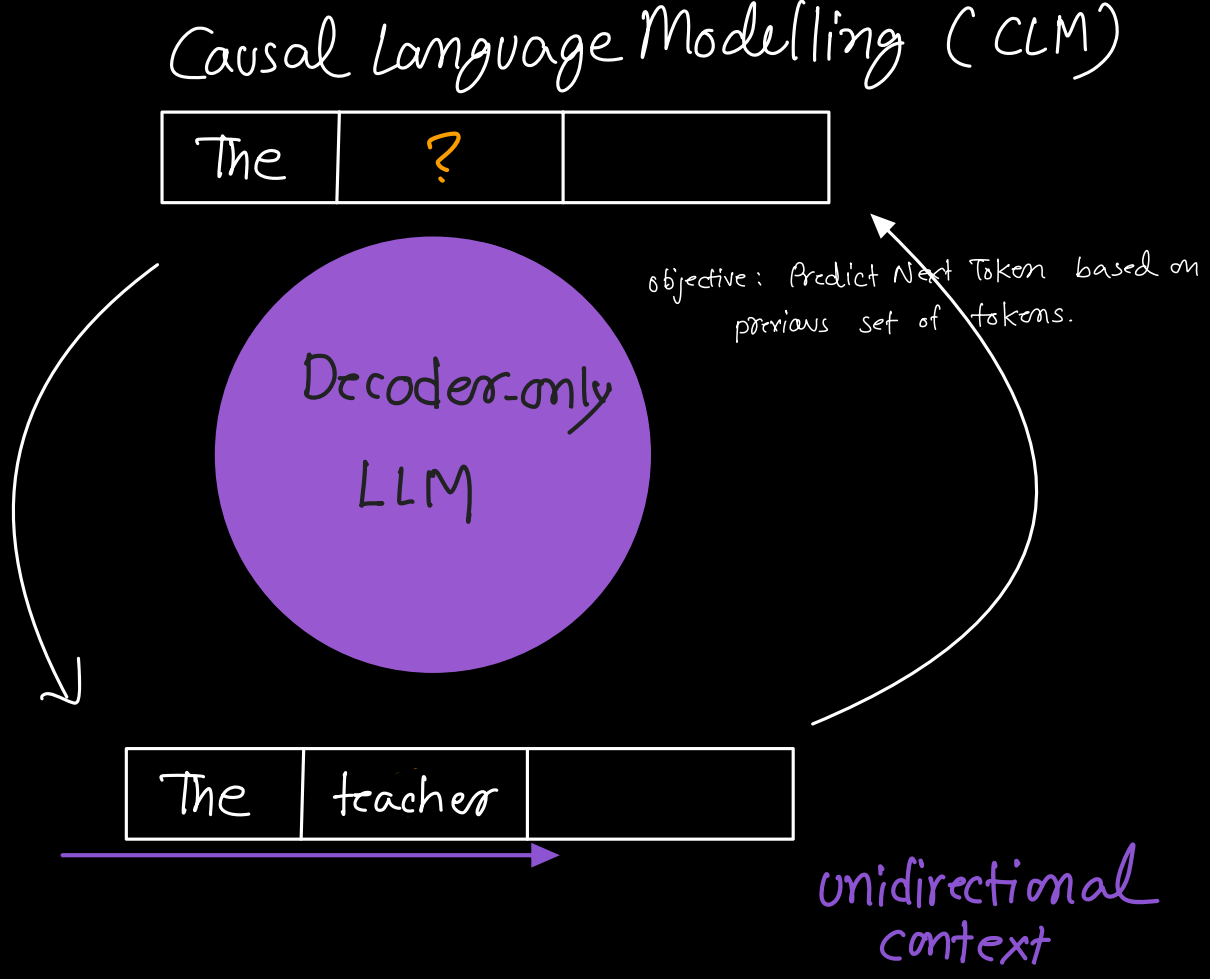
mask

bidirectional

• Decoder only — Autoregressive models



The teacher teaches the student



Good Use cases

- ✓ text generation
- ✓ other emergent behaviour
  - depends on model size

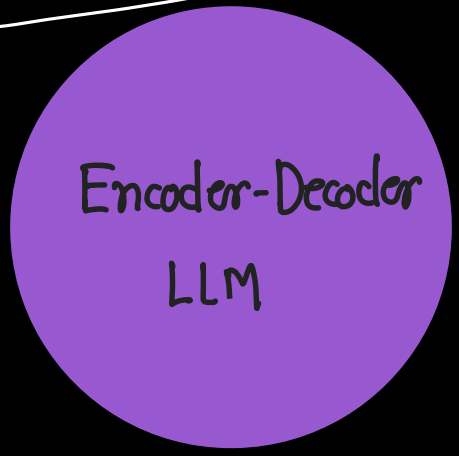
example: GPT
BLOOM

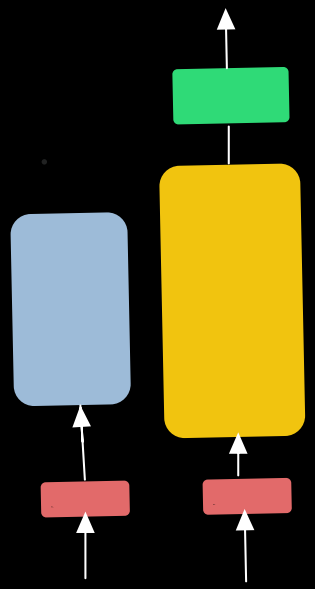
Sequence-to-sequence models

The	teacher	<MASK>	<MASK>	student
The	teacher	<X>		student

sentinal token

special tokens added to the vocabulary





Usecases :

translation

Text summarization

QA

example models

- T5
- BART

Objective : Reconstruct span

<X>	teaches	the

Computational Challenges

when you try to train large language models

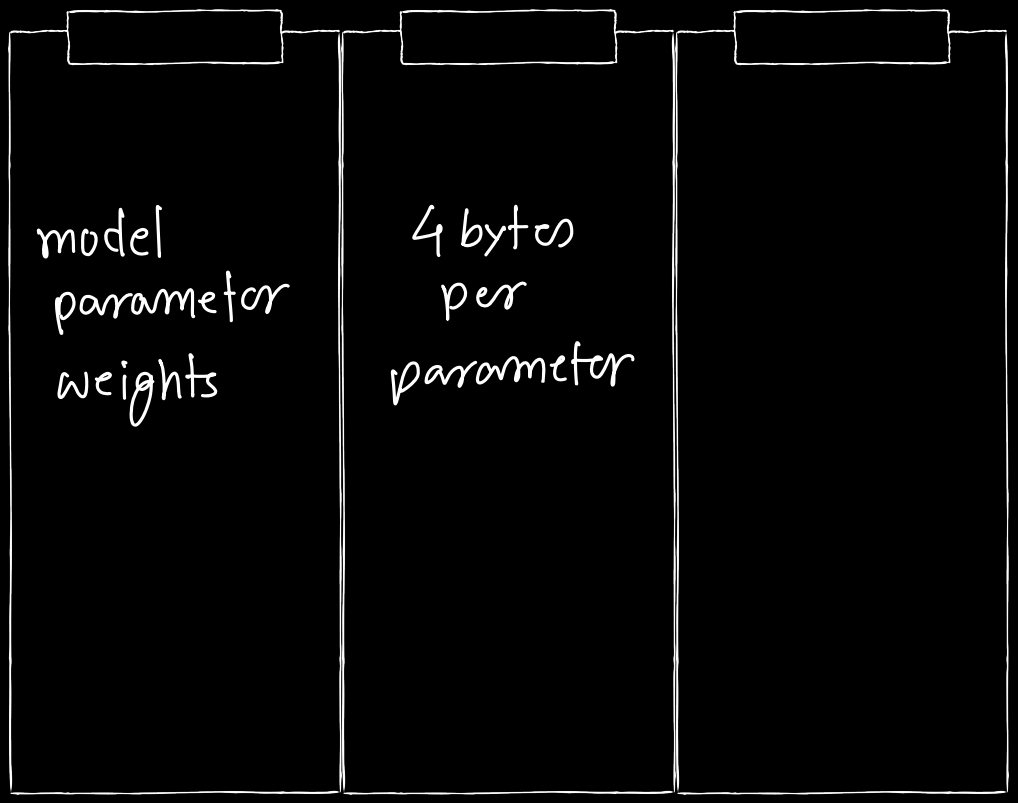
Out of memory : Cuda out of memory.

approximate GPU RAM needed to store 1B parameters

1 parameter = 4 bytes (32-bit float)

1B parameters = 4×10^9 bytes = 4GB

4GB @ 32-bit full Precision



adam optimizers (2 states)	+8 bytes per parameter
Gradients	+ 4 bytes per parameter
Activations and temp memory (variable size)	+8 bytes per parameter (high-end estimate)

} 20 extra bytes per parameter

total = 4bytes per parameter + 20 extra bytes per parameter

memory needed to store the model

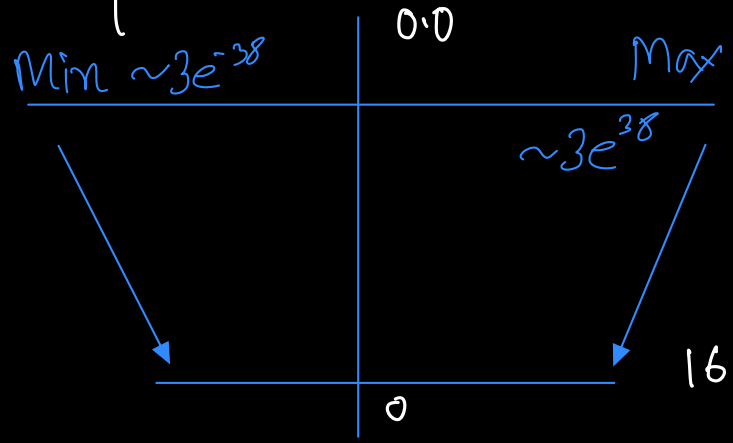
memory needed to train the model

4GB @ 32-bit full Precision

80GB @ 32-bit full precision

How do we reduce the memory utilization by the models?

Quantization



FP 32 4 byte memory

32-bit floating point

Range From ~3e^-38 to ~3e^38

16-bit floating point | 8-bit integer

F16 | BFLOAT16 | INT8

example : Pi 3.141592

FP 32

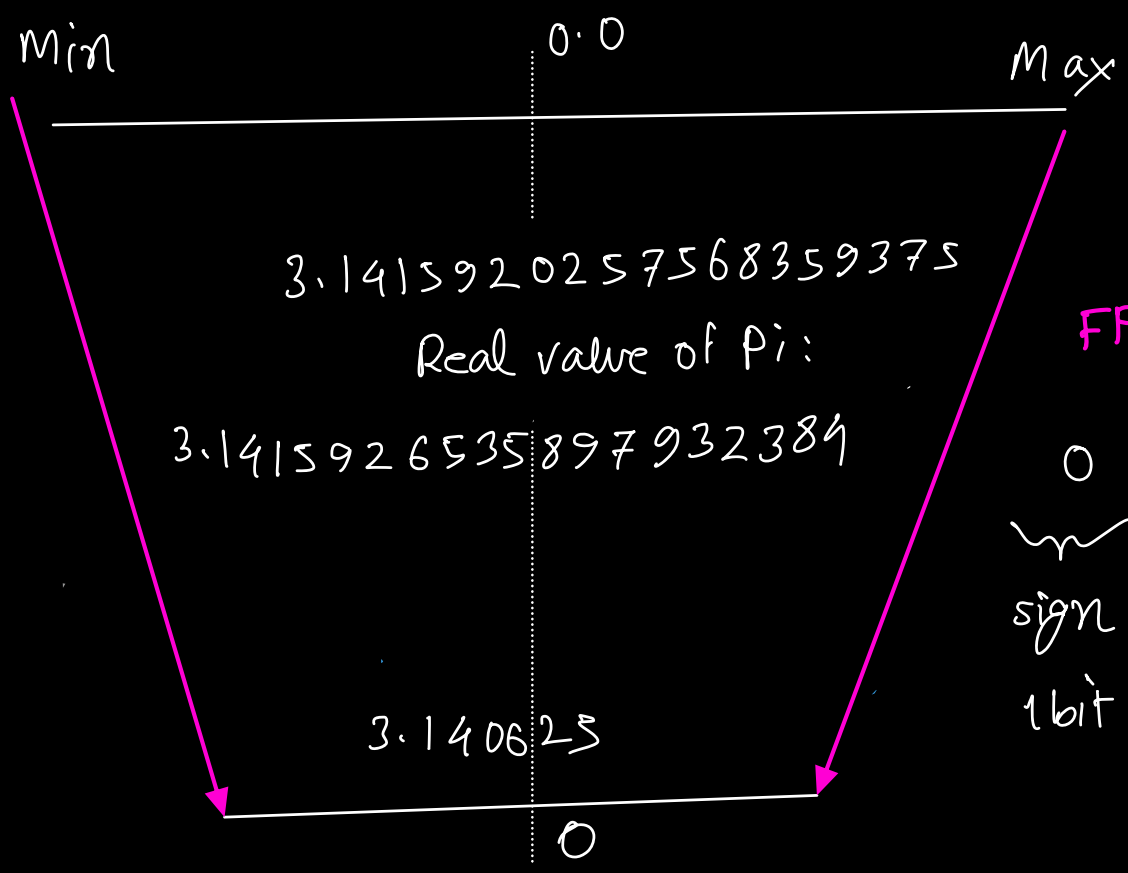
0 | 10000000 | 10010010000111111011000

Sign
1bit

Exponent
8bits

Fraction
23bits

Mantissa / Significand
= Precisim



FP16 2 bytes memory

0 | 10000 | 1001001000

sign
1bit

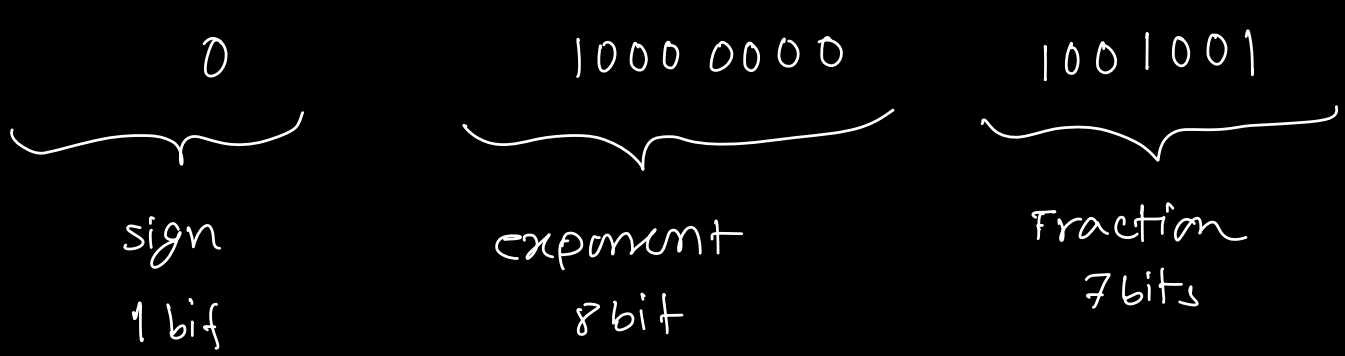
exponent
5bit

fraction
10bit

downside of B-float is not suited for integer calculations

BFLOAT16 | BF16

2 bytes memory



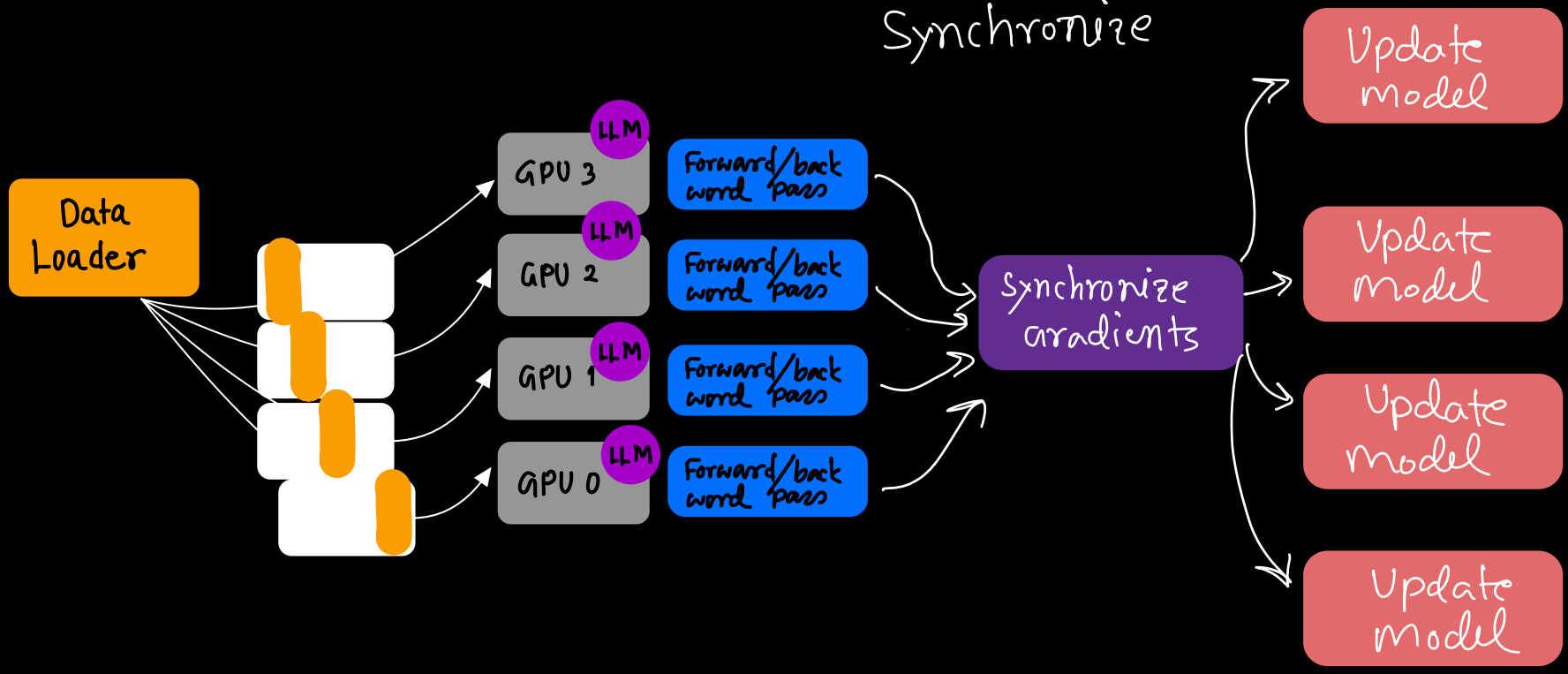
Truncated FP32

- QAT: quantization aware training.

Efficien multi GPU compute strategies

DDP

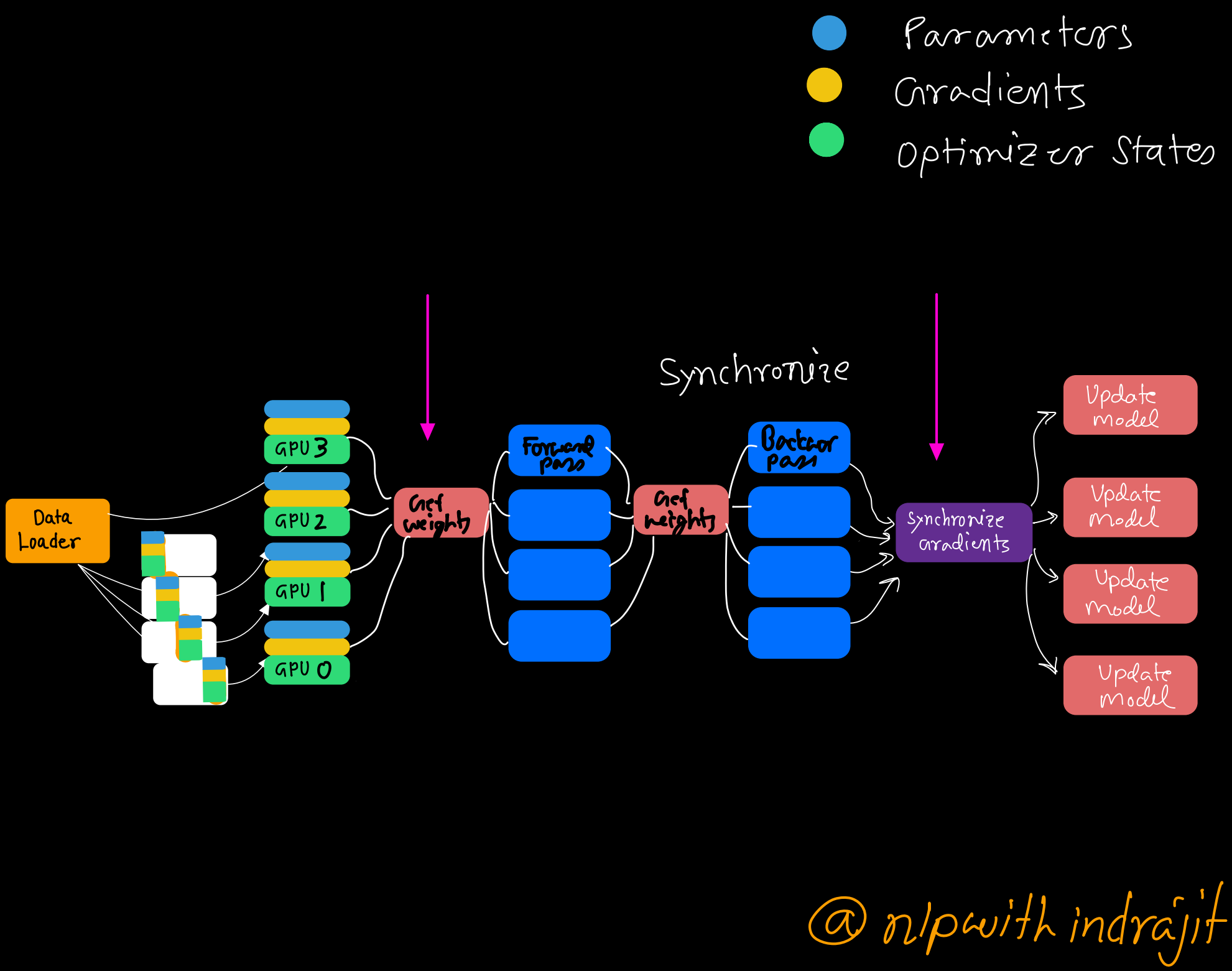
Distributed Data Parallel (DDP)



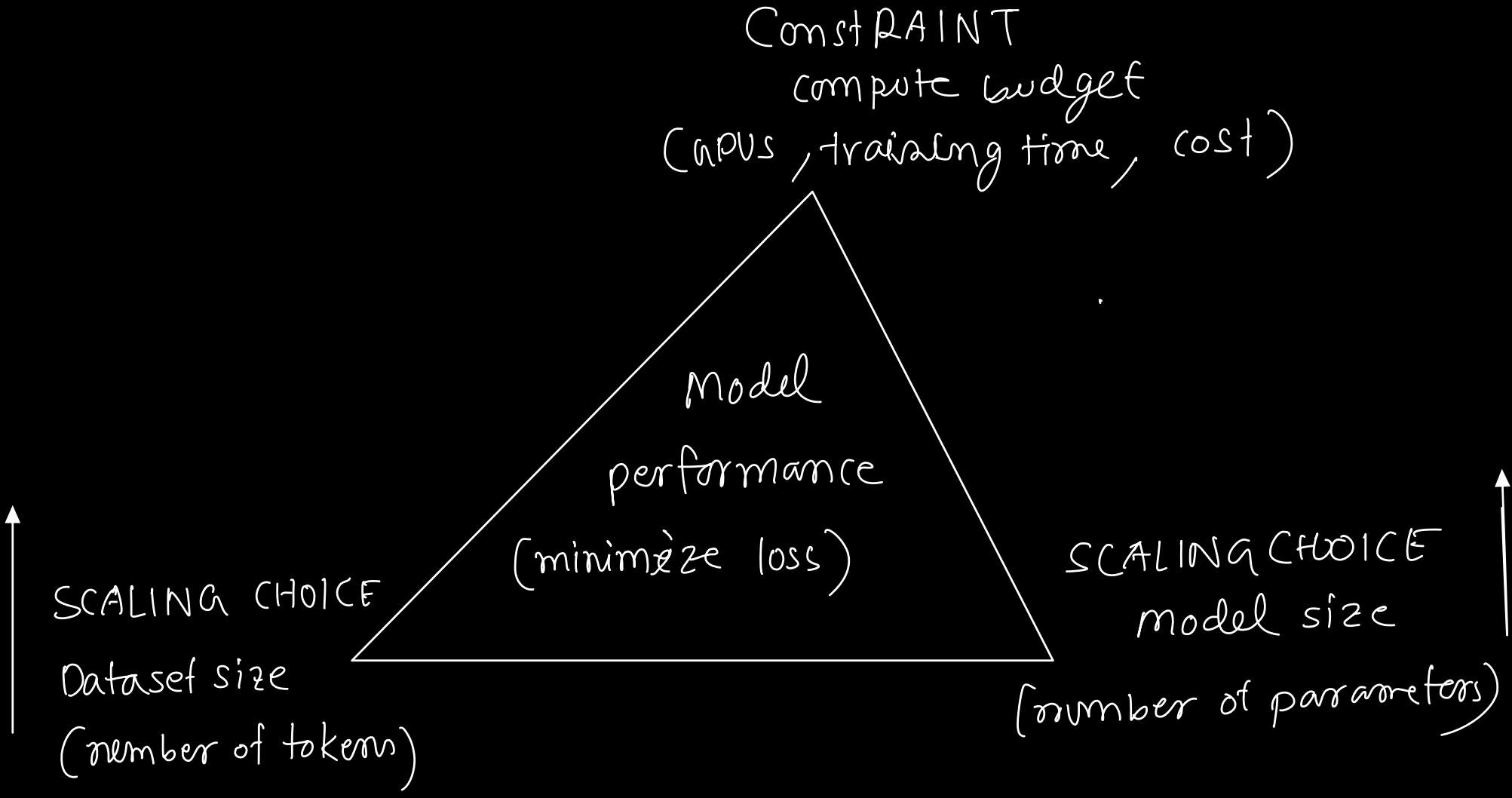
FSDP (Full Sharded Data in Parallel) FSDP

- Motivated by zero paper (ZeRo) - 2019
  - zero data overlap between GPUs

Reduces memory by distributing (sharding) the model parameters, gradients, & optimizer states across GPUs


Parameters
Gradients
Optimizer States
Synchronize
Data Loader
GPU 3
GPU 2
GPU 1
GPU 0
Get weights
Forward pass
Get weights
Backward pass
Synchronize gradients
Update Model
Update Model
Update Model
Update Model
@ nlpwithindrajit

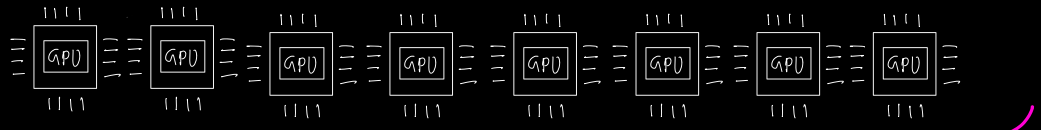
Scaling Laws and compute optimal Models



1 Peta FloP = 1,000,000,000,000,000 (one quadrillion) floating point operations per second.

"1 peta flop/s-day" = floating point operations performed at a rate of 1 petaFlop per second for one day.

NVIDIA V100s



24 hours

- Pre-Training for Domain Adaptation

Week - 2

Fine Tuning LLMs with instruction prompts

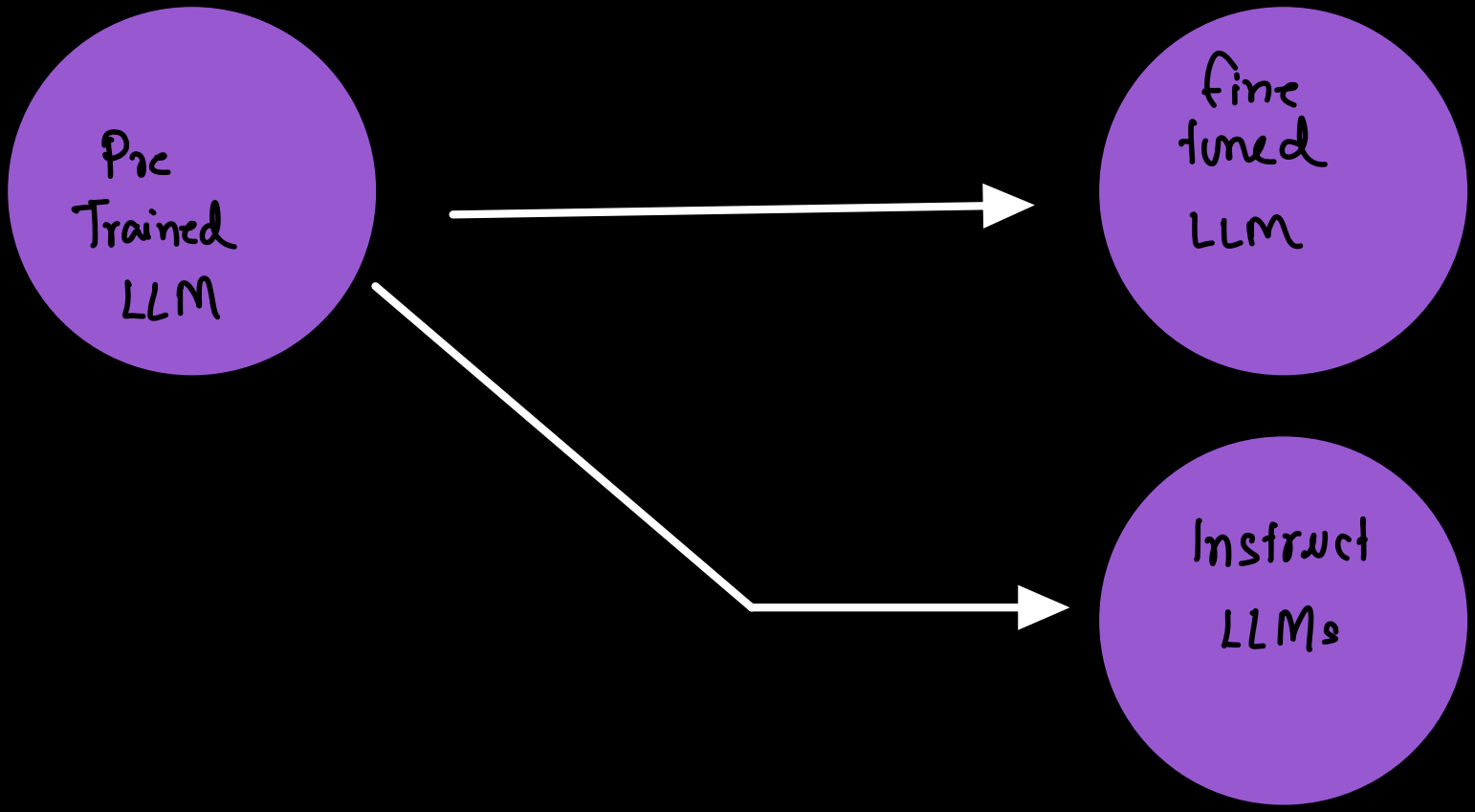
ICL - in context learning (zero shot inference)

Problems

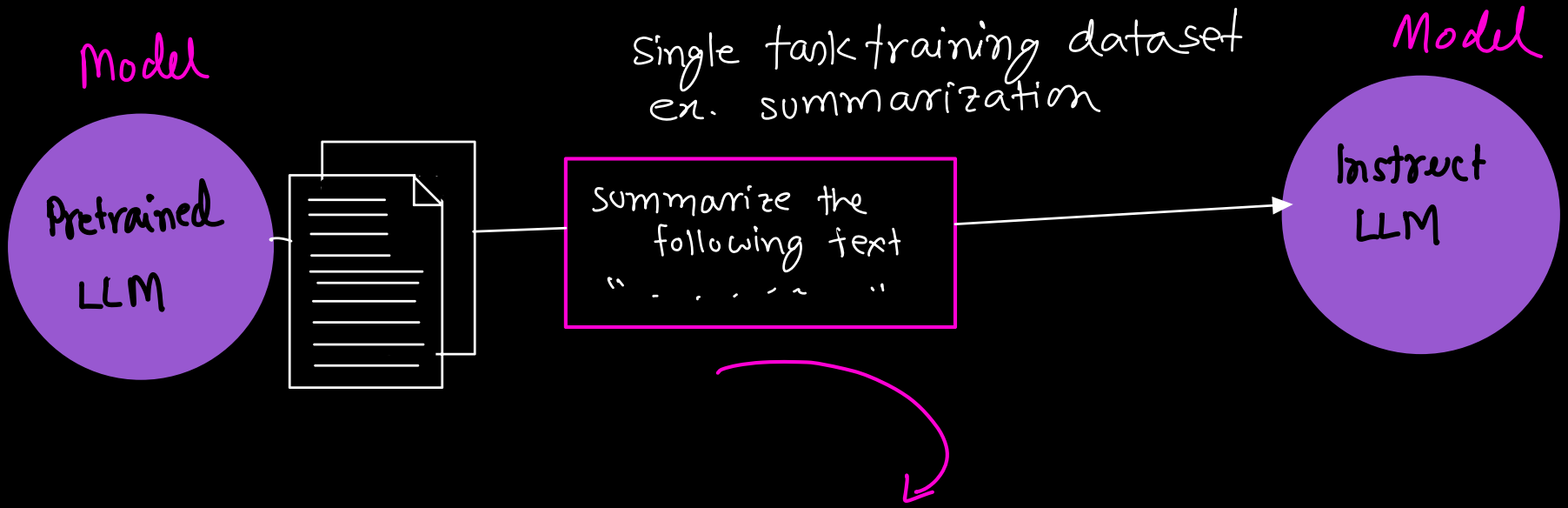
* doesn't work with small LLMs

instruction fine tuning ⟶ full fine tuning

Sample Promt instruction templates



Fine Tuning on a single Task



often needed only 500–1000 examples needed to fine-tune a single task

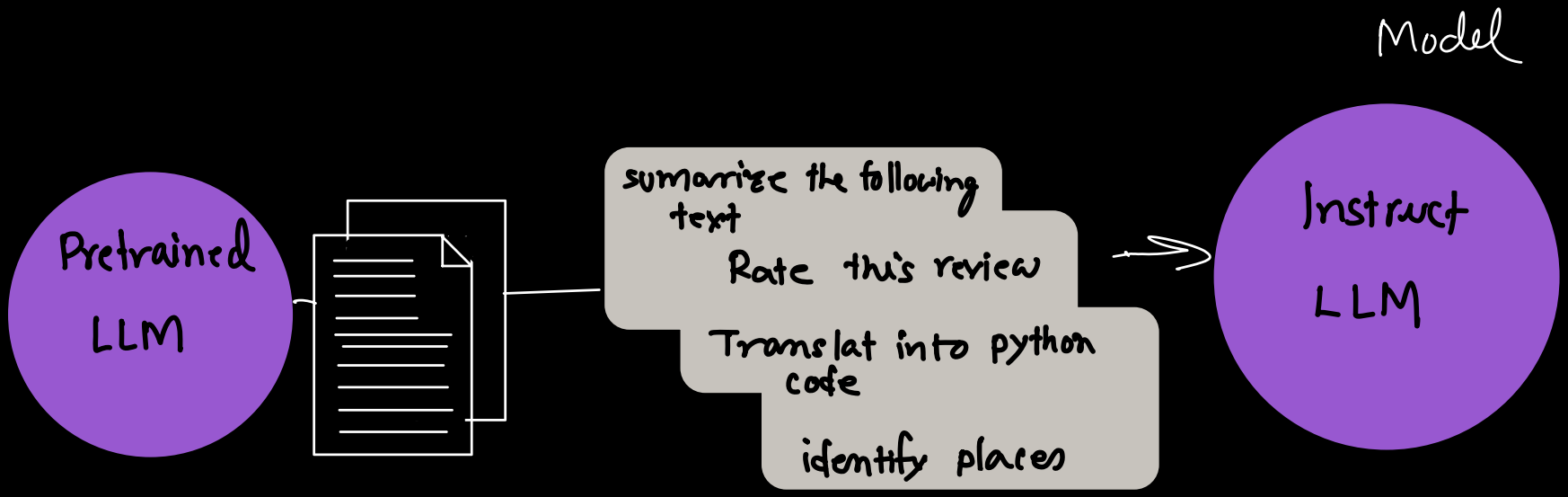
Downside of training on a single task:

→ Catastrophic forgetting

solution

- Fine tune on multiple task at the same time
- PEFT (parameter efficient fine tuning)

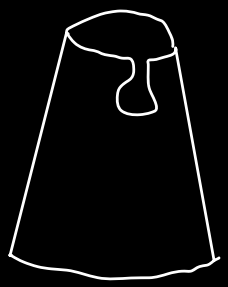
Multi Task fine Tuning



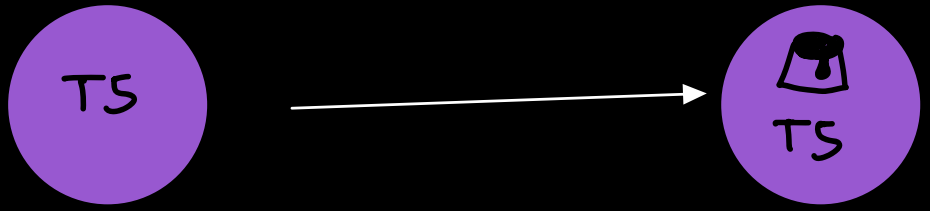
drawback : need a lot of data

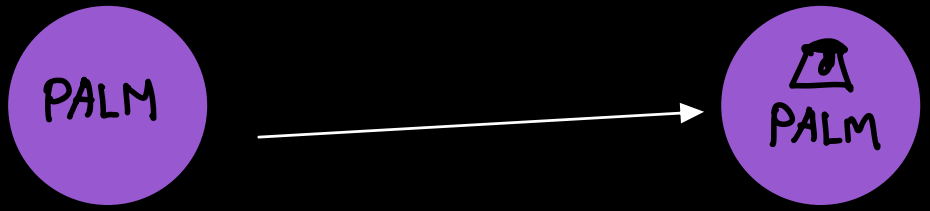
Instruct fine-tuning with FLAN

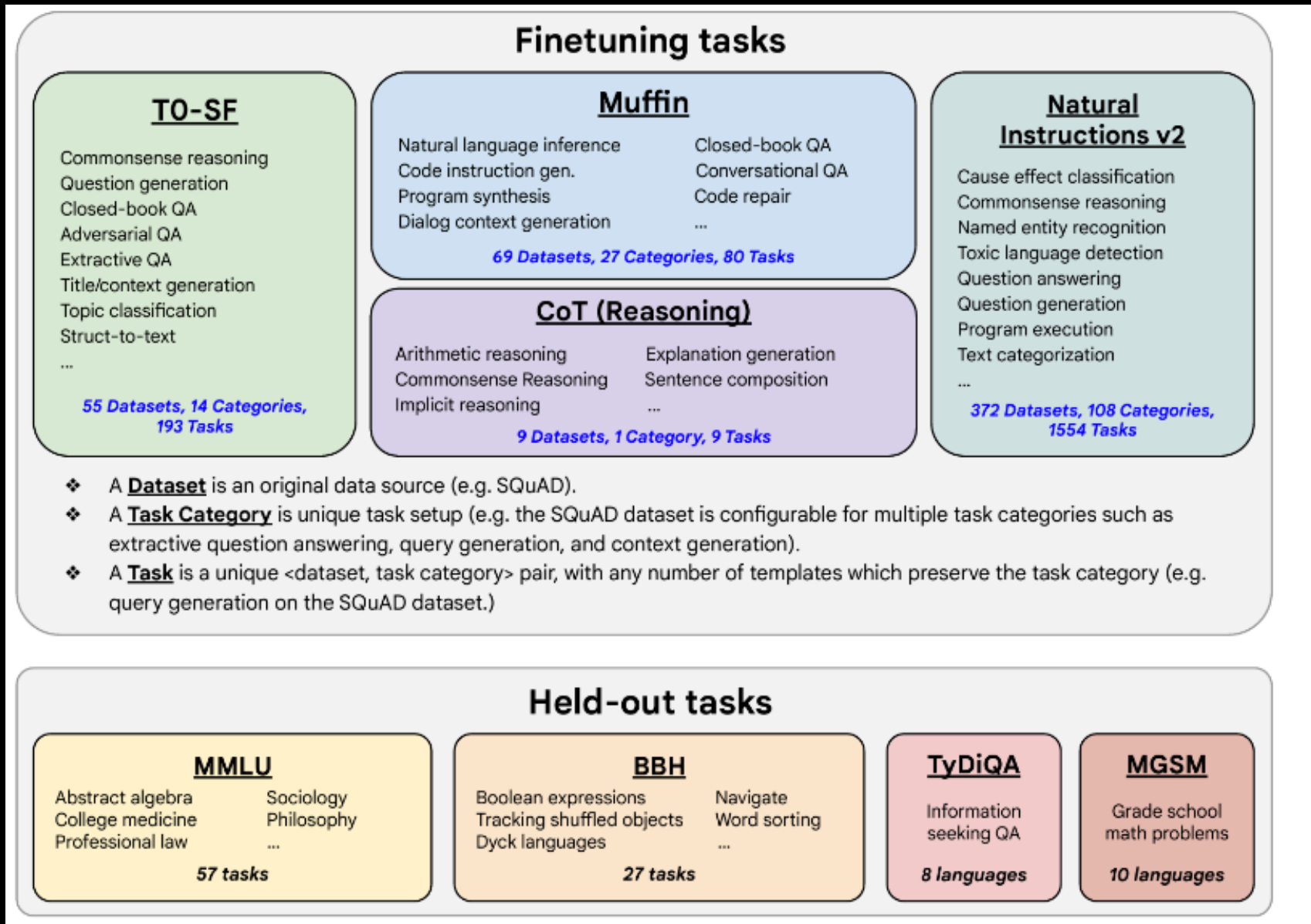
- Flan models refers to a specific set of instructions I used to perform instruction fine-tuning



Fine tuned language net






Finetuning tasks
T0-SF
Commonsense reasoning
Question generation
Closed-book QA
Adversarial QA
Extractive QA
Title/context generation
Topic classification
Struct-to-text
...
55 Datasets, 14 Categories, 193 Tasks
Muffin
Natural language inference
Code instruction gen.
Program synthesis
Dialog context generation
Closed-book QA
Conversational QA
Code repair
...
69 Datasets, 27 Categories, 80 Tasks
CoT (Reasoning)
Arithmetic reasoning
Commonsense Reasoning
Implicit reasoning
Explanation generation
Sentence composition
...
9 Datasets, 1 Category, 9 Tasks
Natural Instructions v2
Cause effect classification
Commonsense reasoning
Named entity recognition
Toxic language detection
Question answering
Question generation
Program execution
Text categorization
...
372 Datasets, 108 Categories, 1554 Tasks
A Dataset is an original data source (e.g. SQuAD).
A Task Category is unique task setup (e.g. the SQuAD dataset is configurable for multiple task categories such as extractive question answering, query generation, and context generation).
A Task is a unique <dataset, task category> pair, with any number of templates which preserve the task category (e.g. query generation on the SQuAD dataset.)
Held-out tasks
MMLU
Abstract algebra
College medicine
Professional law
Sociology
Philosophy
...
57 tasks
BBH
Boolean expressions
Tracking shuffled objects
Dyck languages
Navigate
Word sorting
...
27 tasks
TyDiQA
Information seeking QA
8 languages
MGSM
Grade school math problems
10 languages

LLM evaluation metrics :

It is cold outside

It is not cold outside

Used for text summarization

used for text translation

$$\text{Rouge-1 recall} = \frac{\text{unigram matches}}{\text{unigram in reference}} = \frac{4}{4} = 1.0$$

$$\text{Rouge-1 precision} = \frac{\text{unigram matches}}{\text{unigram in output}} = \frac{4}{5} = 0.8$$

$$\text{Rouge-1 F1} = 2\,\frac{\text{Precision} \times \text{recall}}{\text{Precision} + \text{recall}} = 2 \times \frac{0.8}{1.8} = 0.89$$

LLM evalution Metrics – ROUGE 2

Rouge-2

It is cold outside

It is | is cold | cold outside

It is very cold outside

It is | is very | very cold | cold outside

$$\text{ROUGE-2 Recall} := \frac{\text{bigram matches}}{\text{bigram in reference}} = \frac{2}{3} = 0.67$$

$$\text{ROUGE-2 precision} := \frac{\text{bigram matches}}{\text{bi gram in output}} = \frac{2}{4} = 0.5$$

$$\text{ROUGE-2 F1} := 2\,\frac{\text{Precisim} \times \text{recall}}{\text{Precisim} + \text{recall}} = 2 \times \frac{0.335}{1.17} = 0.57$$

LLM Metrics = Metrics - ROUGE-L

It is cold outsize

It is very cold outside

Longest common subsequence (LCS)

It is

cold outside

2

@nlpwithindrajit

$$\text{ROUGE-L Recall} = \frac{\text{LCS(Gen, Ref)}}{\text{unigrams in reference}} = \frac{2}{4} = 0.5$$

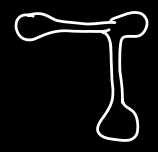
$$\text{ROUGE-L precision} = \frac{\text{LCS(Gen, Ref)}}{\text{unigrams in output}} = \frac{2}{5} = 0.4$$

$$\text{ROUGE-L F1 score} = 2\frac{\text{Precision} \times \text{recall}}{\text{Precision} + \text{recall}} = 2 \times \frac{0.2}{0.9}$$

$$= 0.44$$

evalution metrics benchmarks

GLUE



SUPER GLUE

Benchmarks for massive models

MMLU

Big bench Hard

Big bench Lite

HELM

Holistic evalution of language models.

PEFT

Parameter Efficient fine Tuning

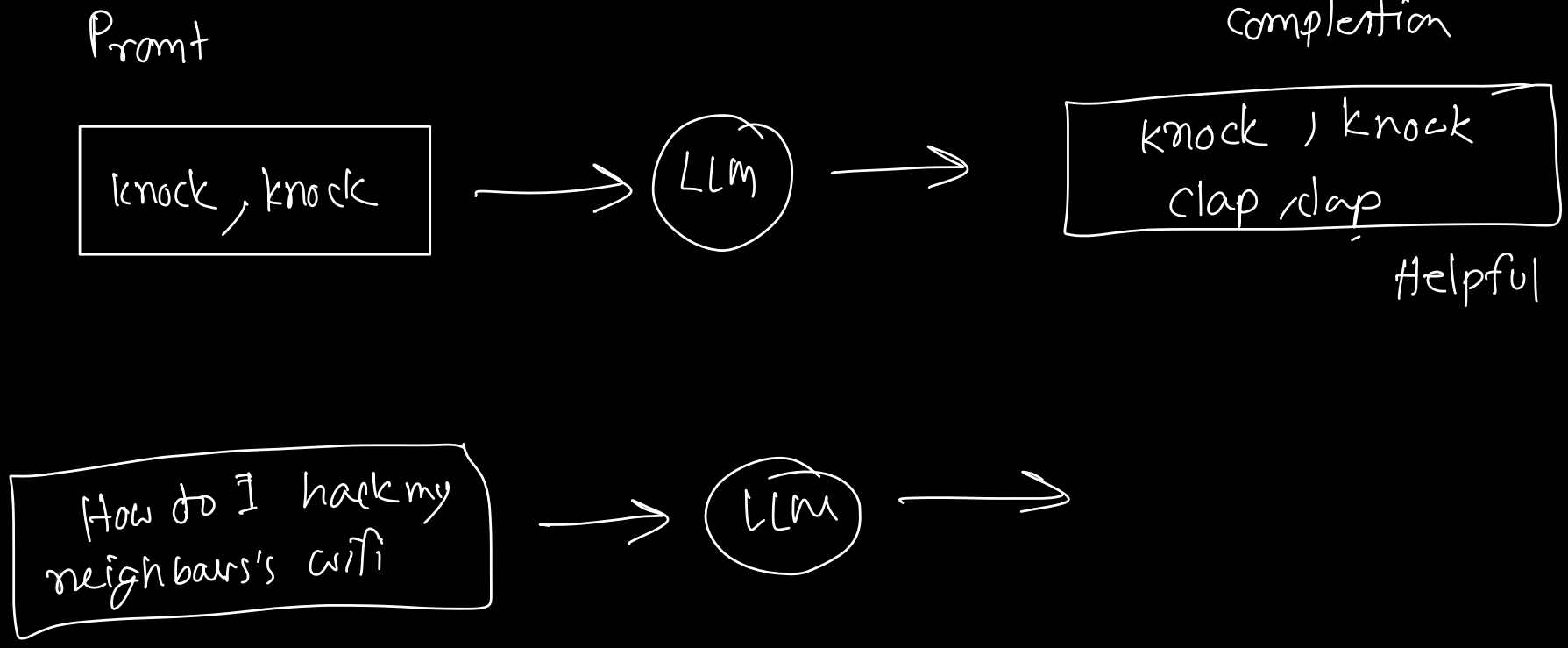
RLHF

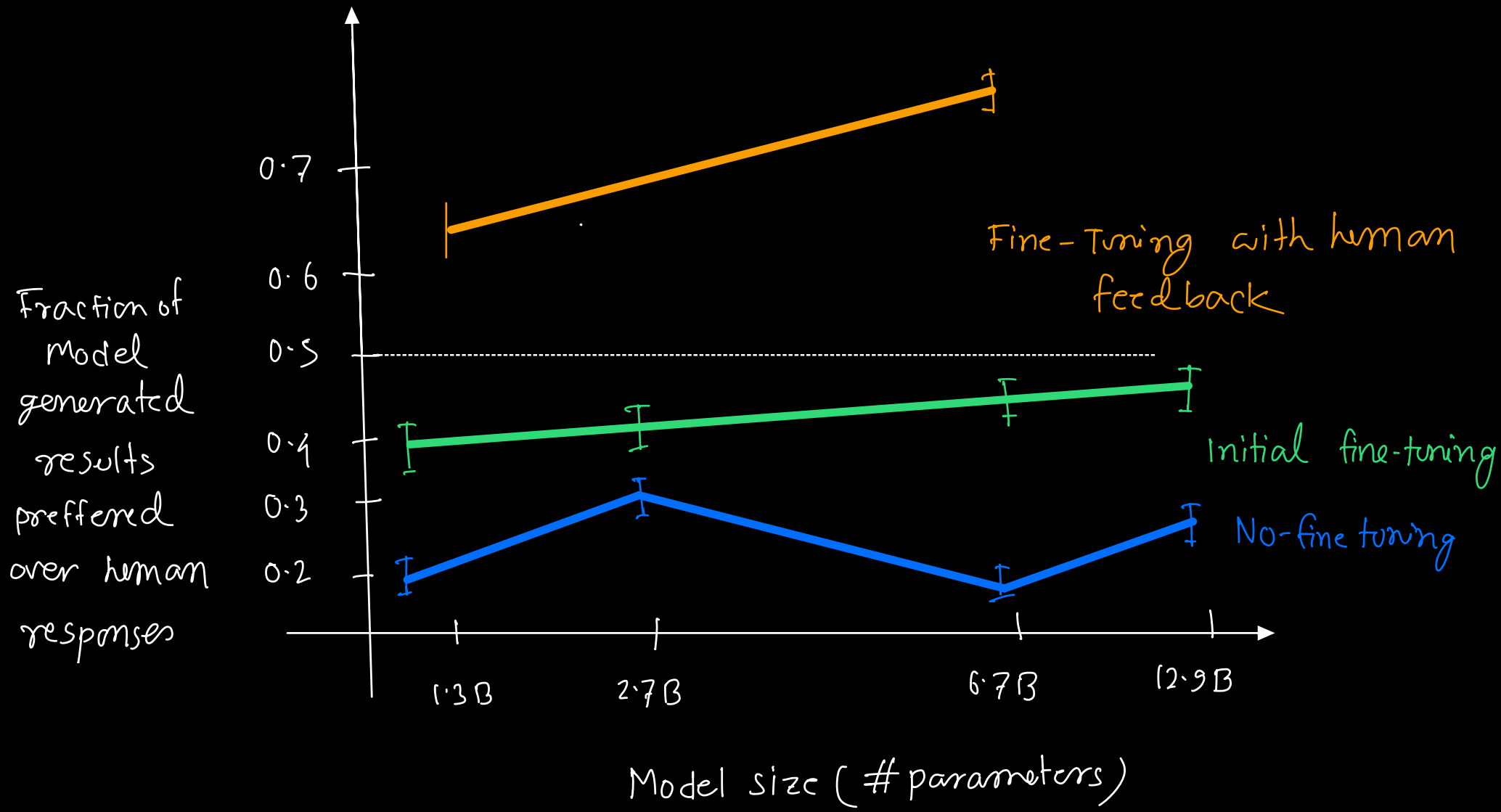
aligning model with human value

Reinforcement Learning with Human Feedback

Aligning models with human values:

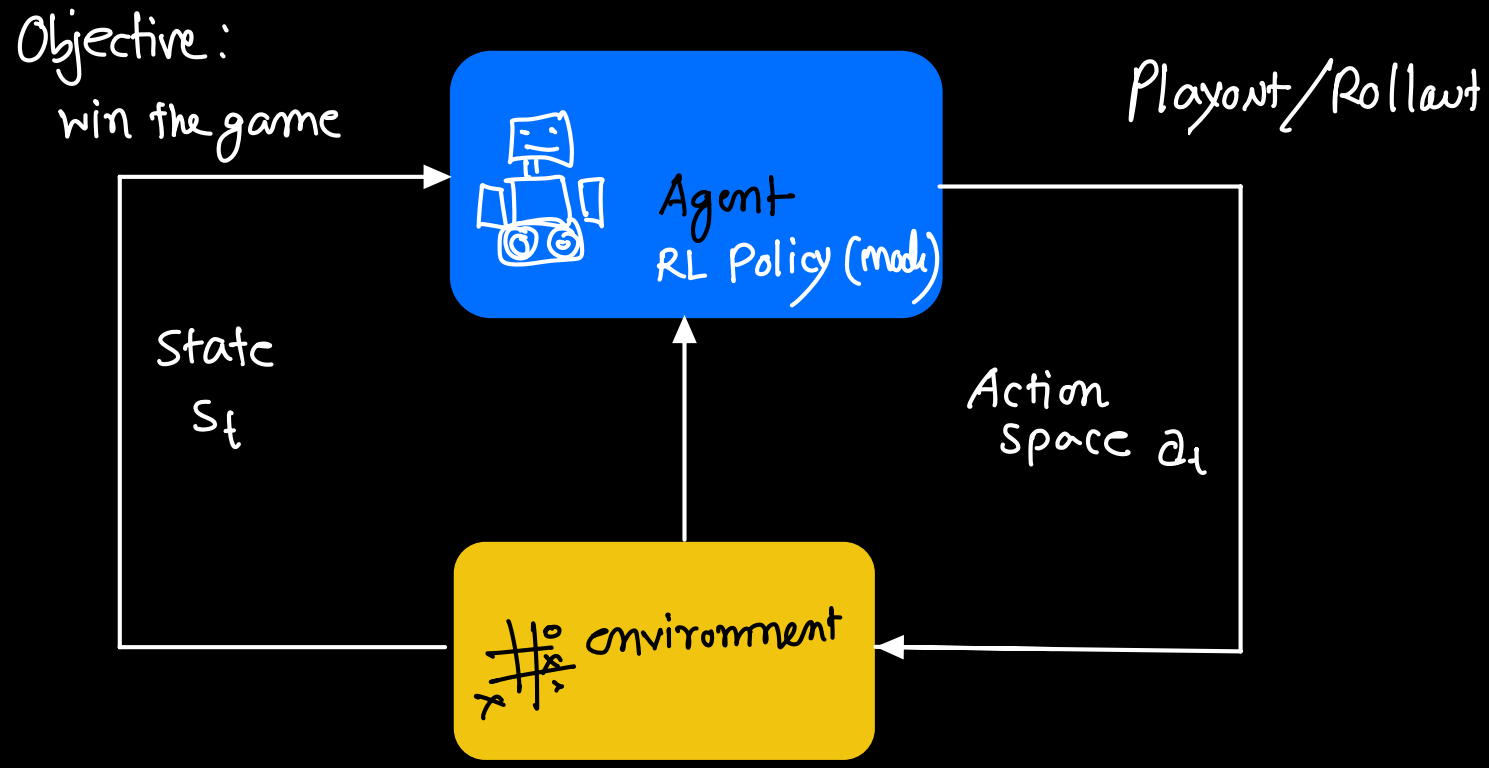
- Models behaving badly
  - Toxic language
  - Aggressive Responses
  - Providing dangerous information


Prompt
Completion
knock, knock
LLM
knock knock
clap clap
Helpful
How do I hack my neighbors's wifi
LLM


Fraction of model generated results preffered over human responses
0·7
0·6
0·5
0·4
0·3
0·2
Fine-Tuning with human feedback
Initial fine-tuning
No-fine tuning
1·3B
2·7B
6·7B
12·9B
Model size (# parameters)

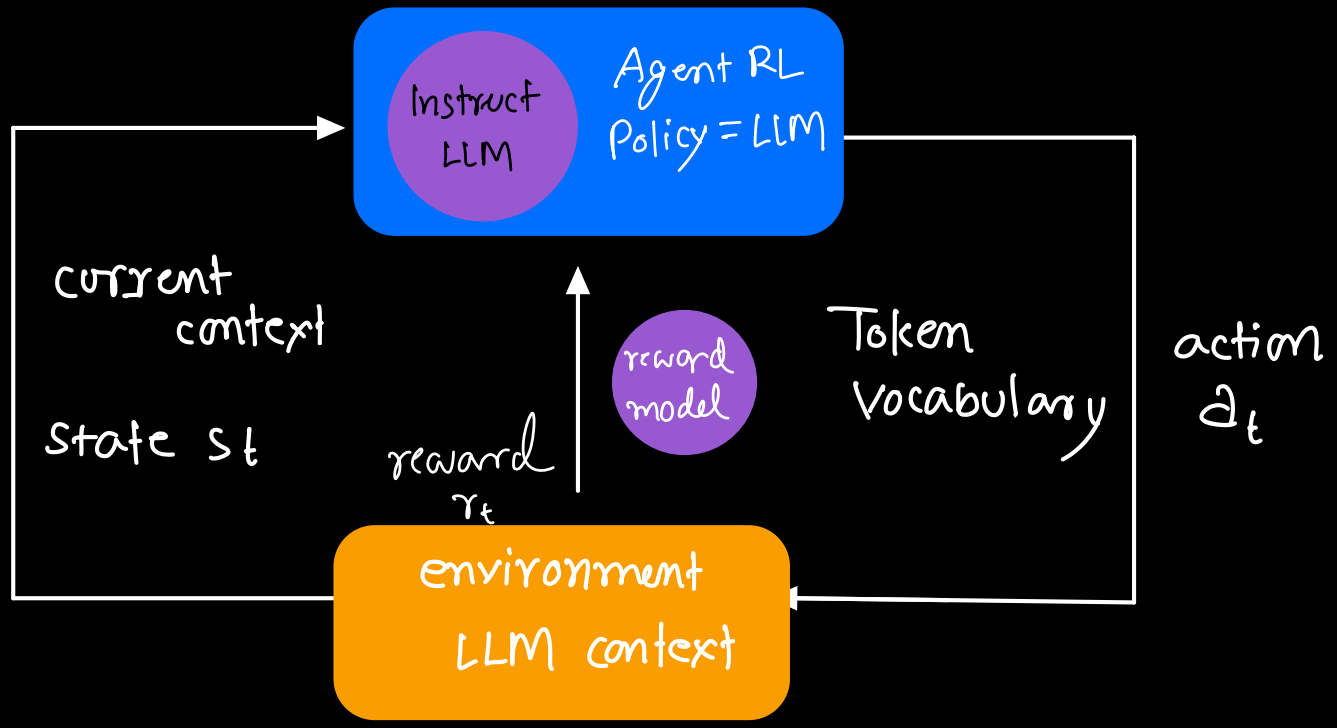
RLHF

Tic-tac-toe game



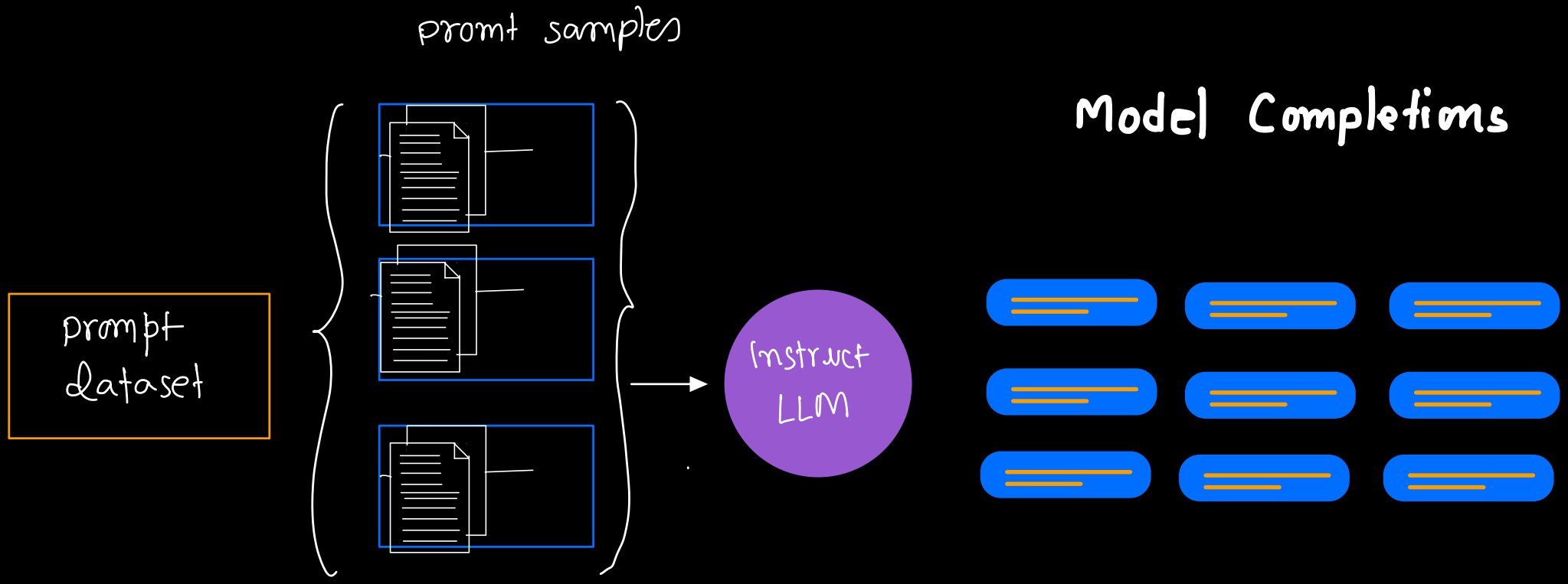
agent make decision based on a strategy known as (RL policy)

@ nlpwith indrajit


Instruct LLM
Agent RL
Policy = LLM
current context
State s_t
reward r_t
reward model
Token Vocabulary
action a_t
environment
LLM context

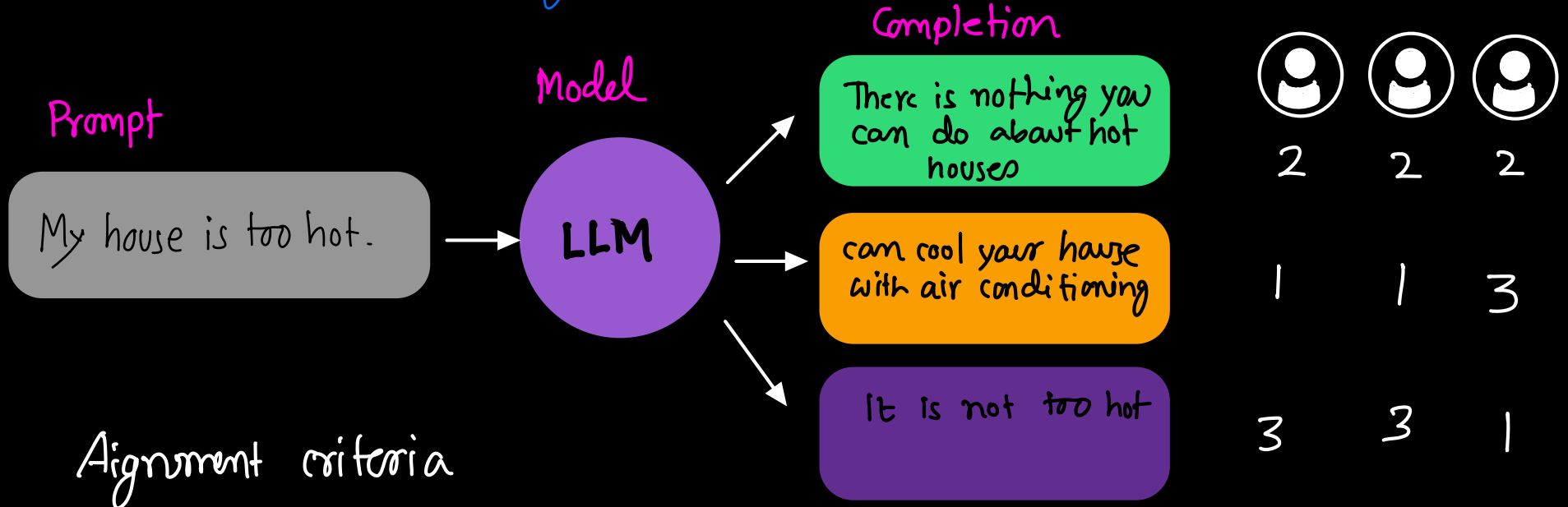
Obtaining feedback from Humans

prepare a dataset from human feedback



Collect human feedback

- Define your model alignment criteria
- For the prompt-response sets that you just generated, obtain human feedback through labeler workforce

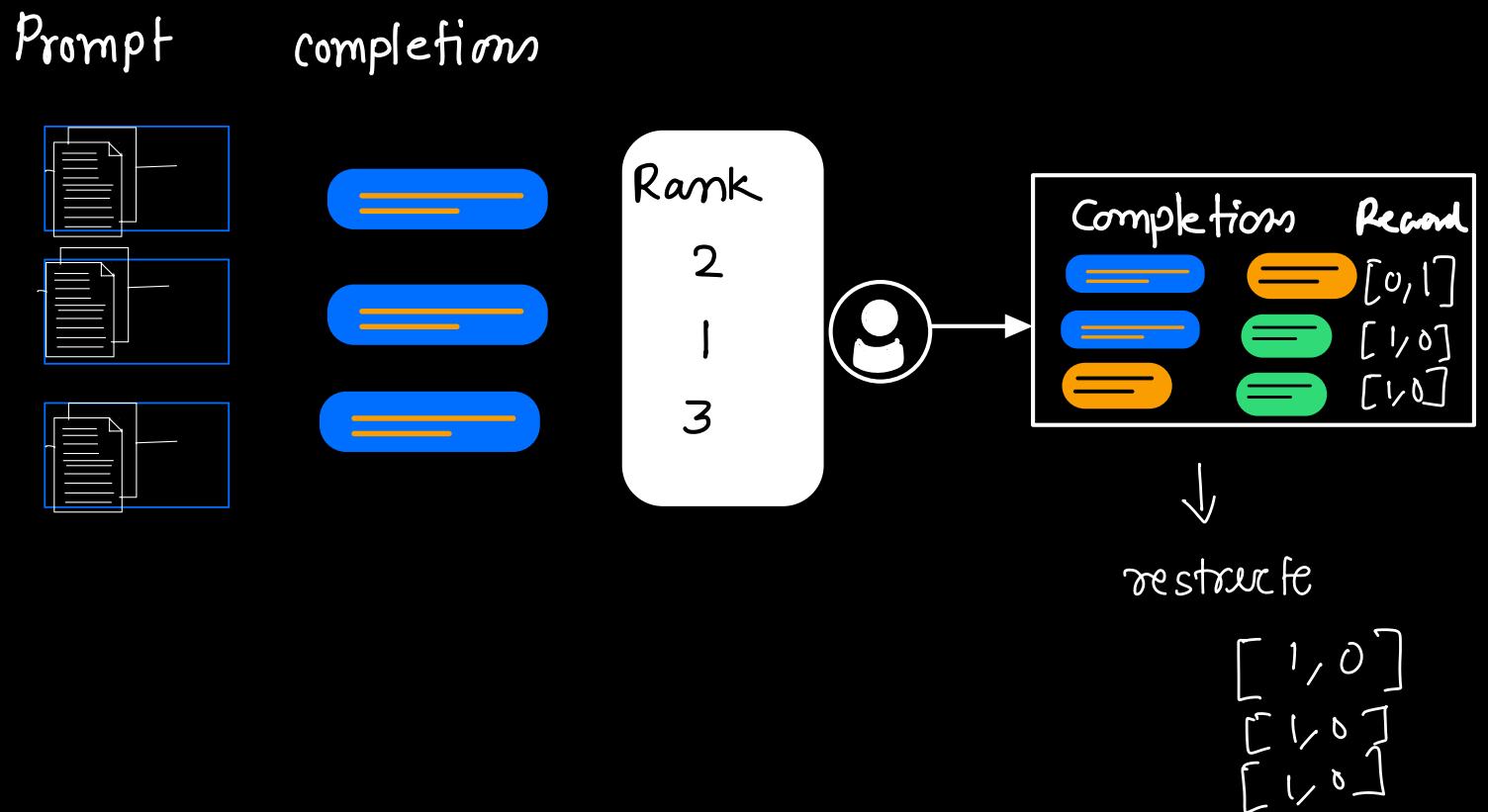


Aignment criteria
: helpfulness

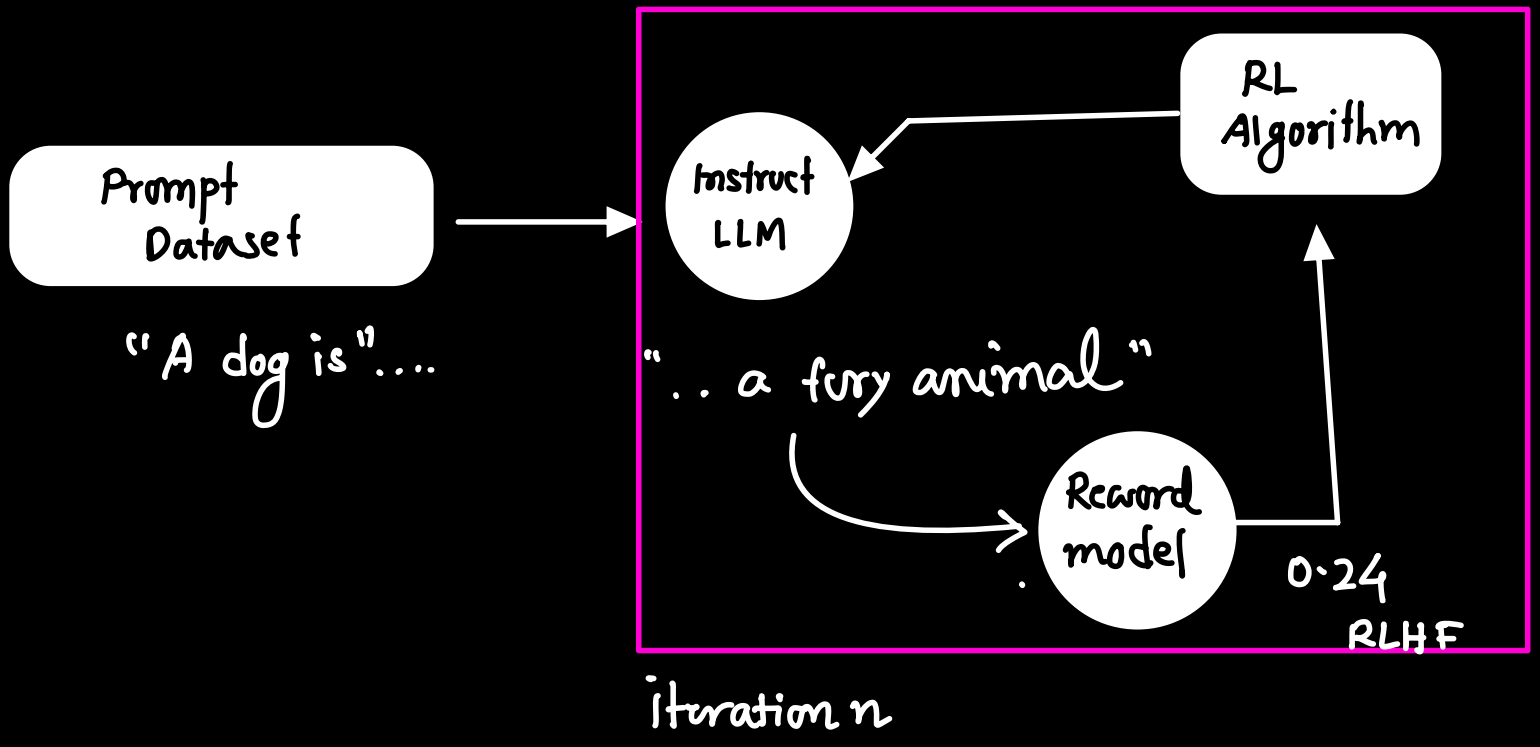
① There should be proper instructions for the human labelers.

② once completed labeling you can train your reward model

③ Convert rankings to pairwise training data for the reward model

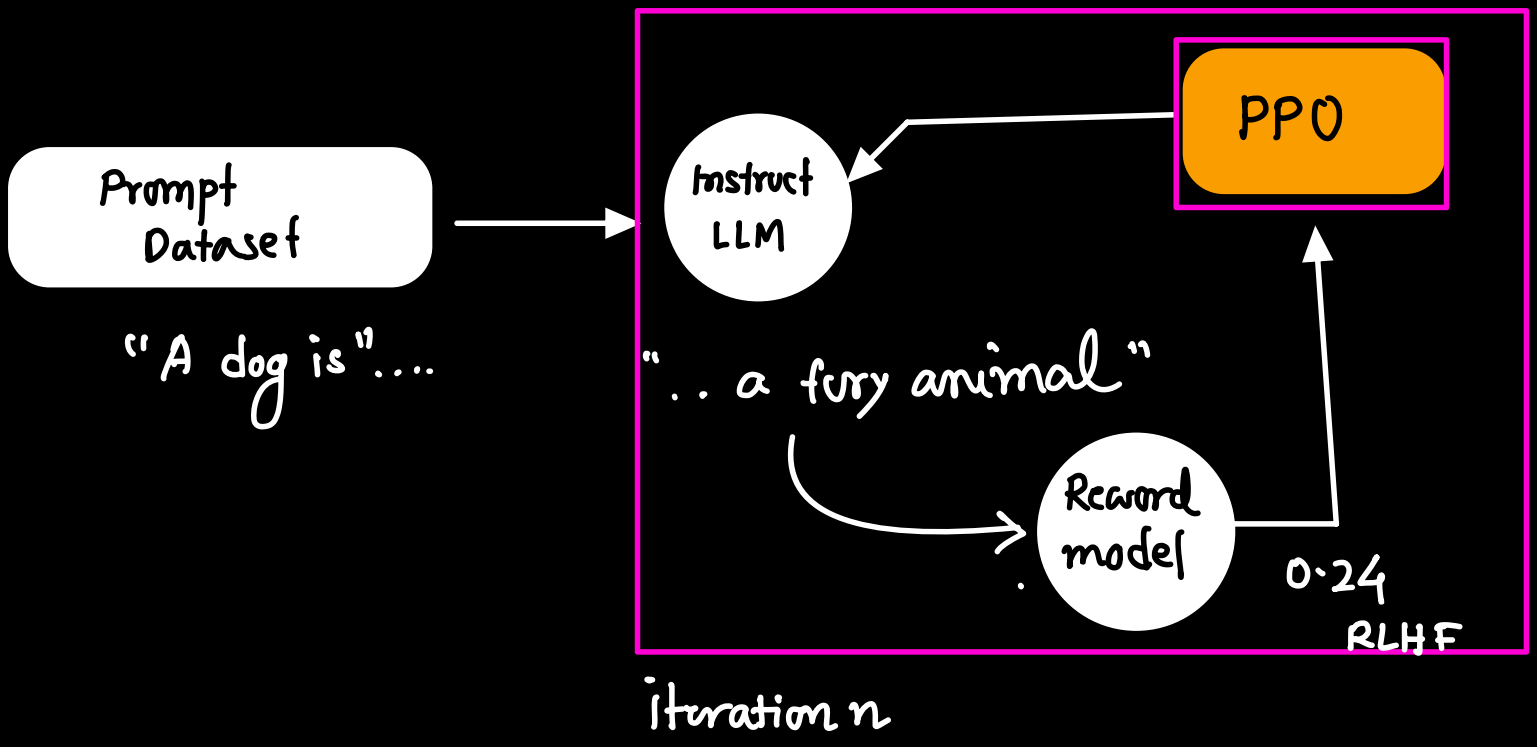


Use the reward model to fine-tune LLM with RL



@ nlpwith indrajit

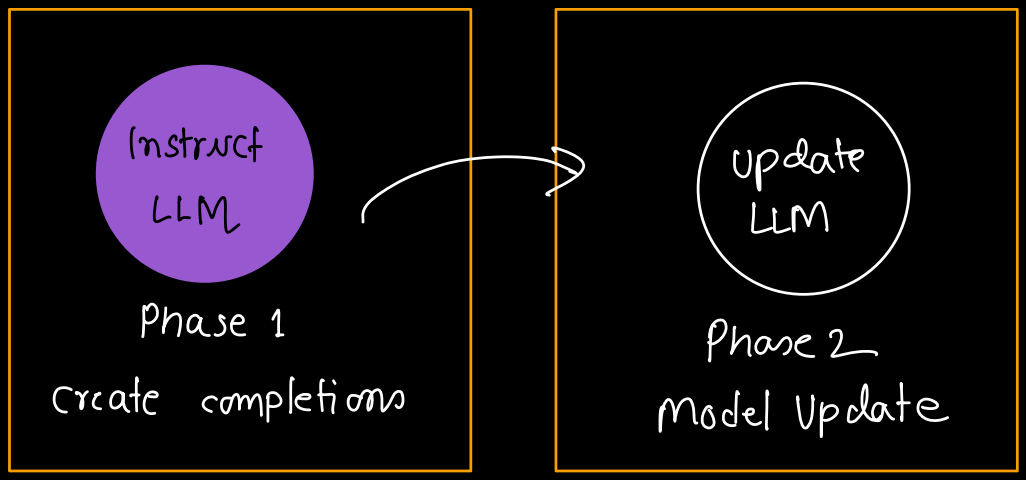
Proximal Policy Optimization



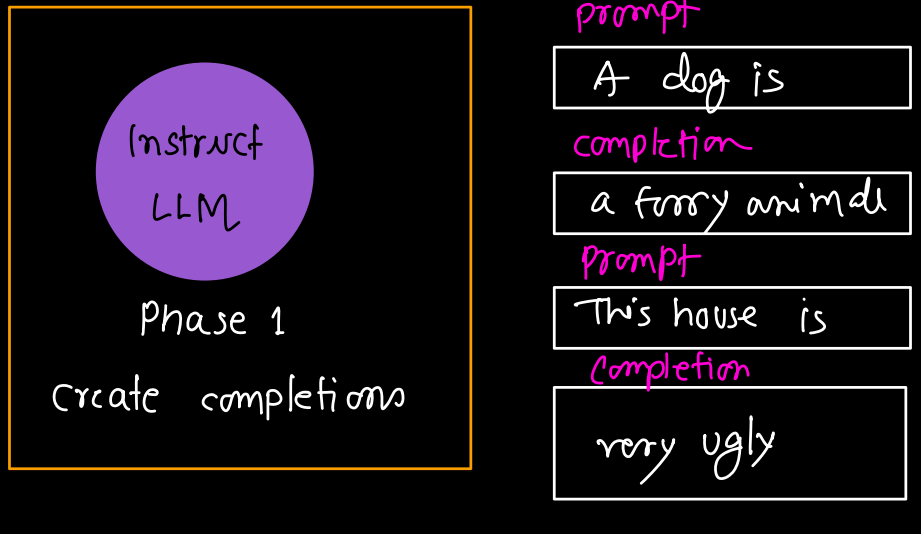
PPO ⟶ optimizes a policy

goal is to update the policy so that the reward is maximized

Initialize PPO with instruct LLM



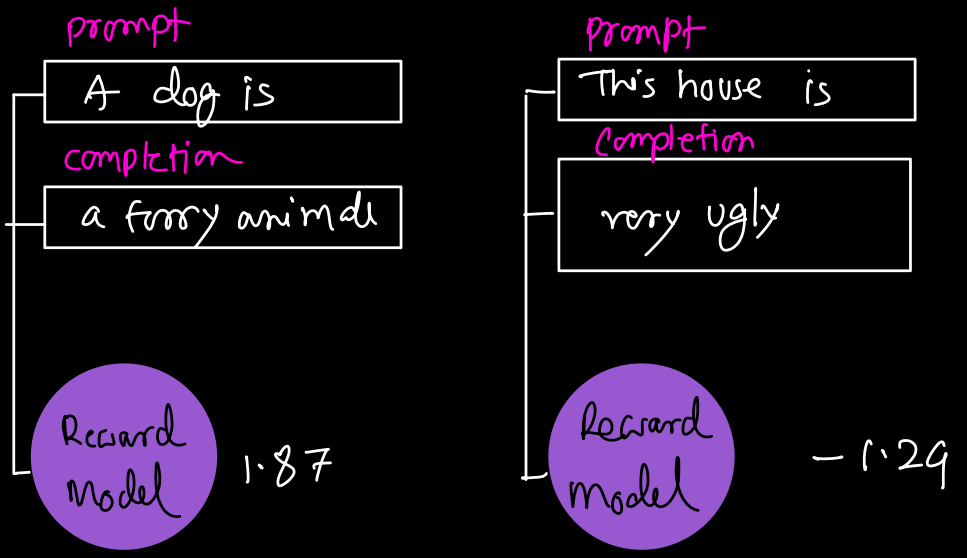
In phase 1 the LLM carries out a number of experiments



Experiments
to assess the outcome of the current model
e.g how harmful, harmless honest the model is

Let's have a closer look at a value function

calculate rewards



value function

$$L^{VF} = \frac{1}{2} \left|\left| V_\theta(s) - (\sum_{t=0}^{T} y^t r_t | s_0 = s) \right|\right|_2^2$$

$V_\theta(s)$: estimated future total reward — 0.34

$(\sum_{t=0}^{T} y^t r_t | s_0 = s)$: known future total reward — 1.87

next generated token = 1.23

PPO Phase 2 : Calculate policy loss

probablities of the next token with the updated LLM

$$L^{policy} = \min\left(\frac{\pi_\theta(at|St)}{\pi_{O_{01d}}(at|St)} \cdot \hat{A}_t\right)$$

probablities of the next token with the initial LLM

advantage term

why ?

positive advantage indicate the suggested token is better than the average

i.e increasing the probablity of the current token leads to advantage.

Defines "Trust region"

$$L^{POLICY} = \min\left(\frac{\pi_\theta(at|St)}{\pi_{O_{01d}}(at|St)} \cdot \widehat{A}_t \,,\, \text{clip}\left(\frac{\pi_\theta(at|St)}{\pi_{O_{01d}}(at|St)}, 1-\varepsilon, 1+\varepsilon\right) \cdot \widehat{A}_t\right)$$

Guardrails

"keeping the policy in the trust region"

$$L^{ENT} = cmtropy\left(\pi_\theta\left(\cdot|s_t\right)\right)$$

similar to temperature setting

If you keep entropy low: you might end up always completing the prompt on the same way.

High entropy: guides the llm to more creativity

PPO : objective function

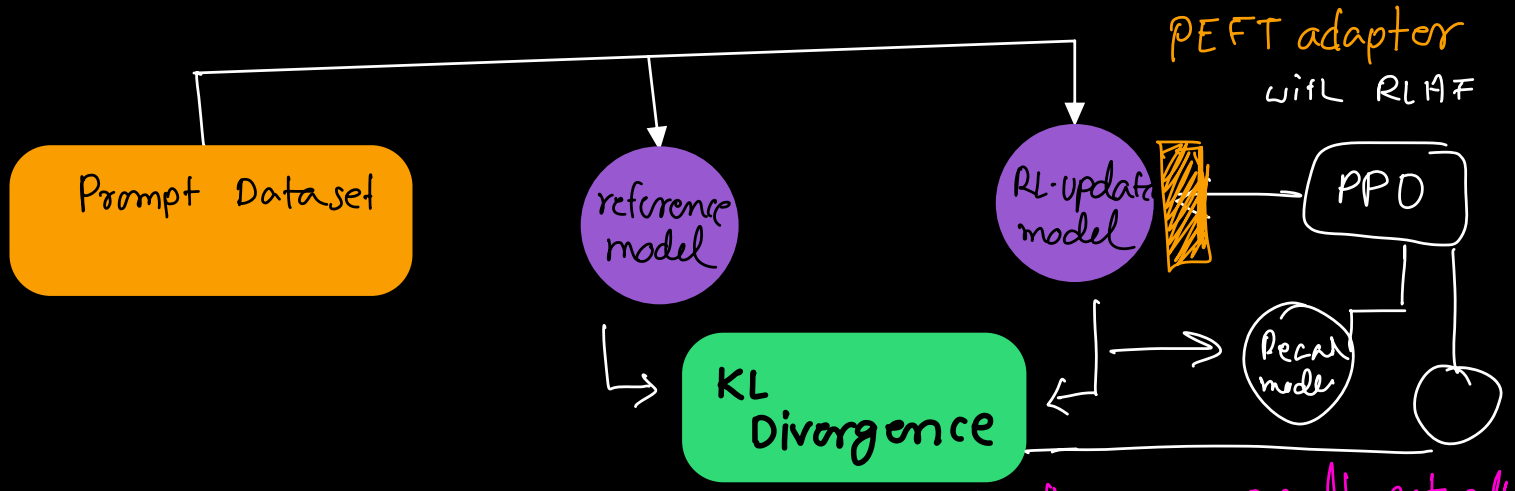
Hyperparameters

$$L^{PPO} = L^{POLICY} + C_1 L^{VF} + c_2 \lfloor^{ENT}$$

policy loss — value loss — Entropy loss

once it completes one iteration PPO, it starts another iteration of PPO and so on.

Avoiding reward hacking can use



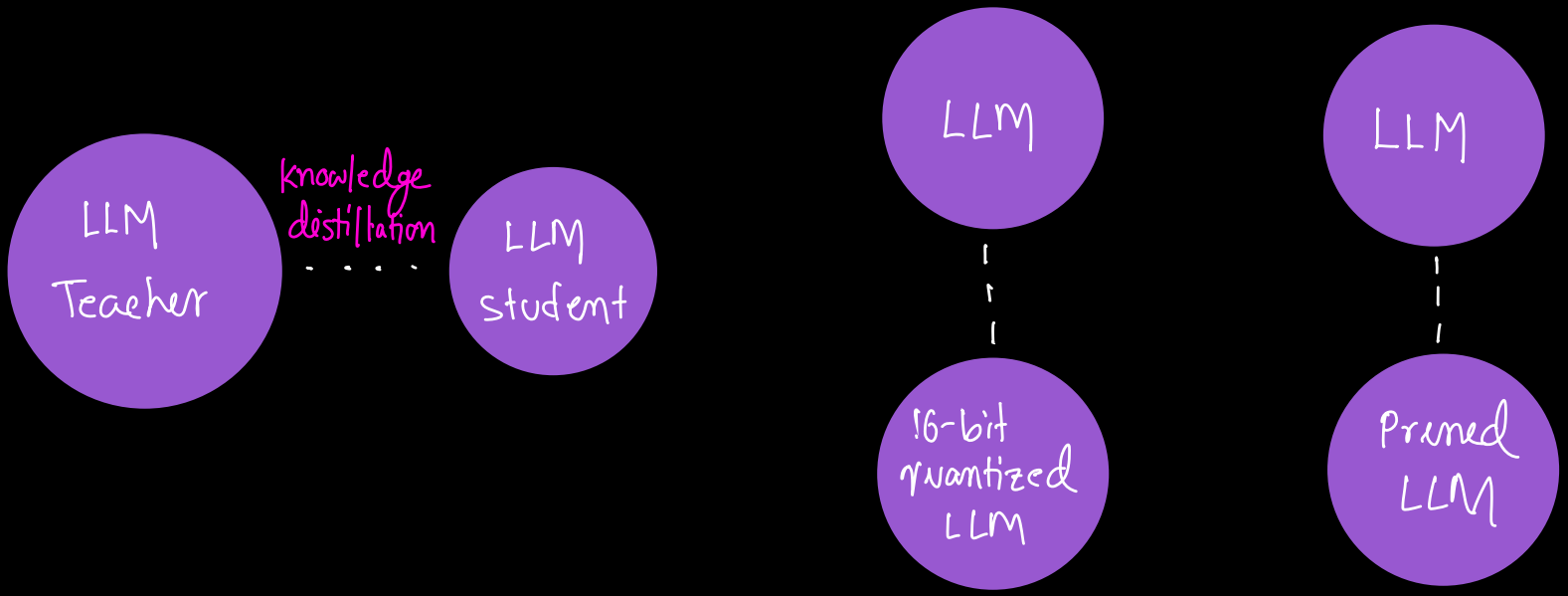
LLM Powered Applications

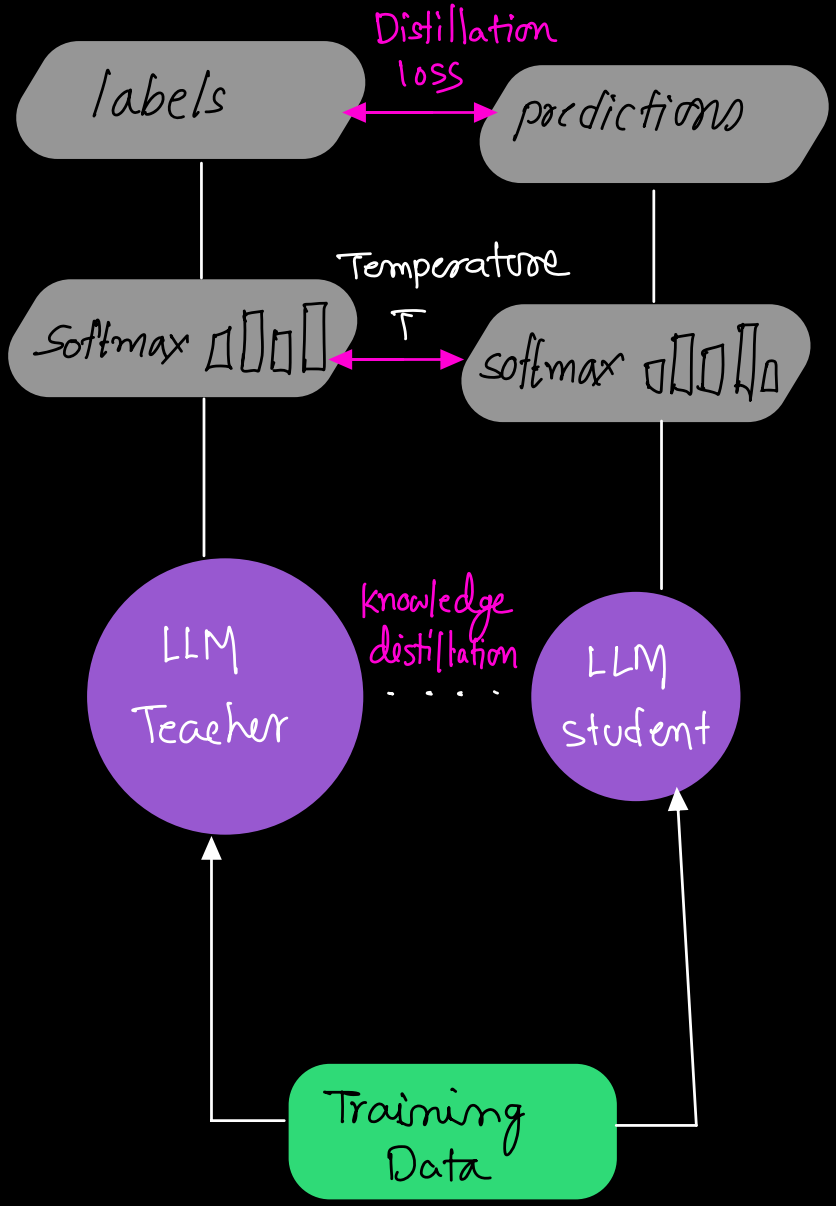
LLM Optimization Technique

Distillation

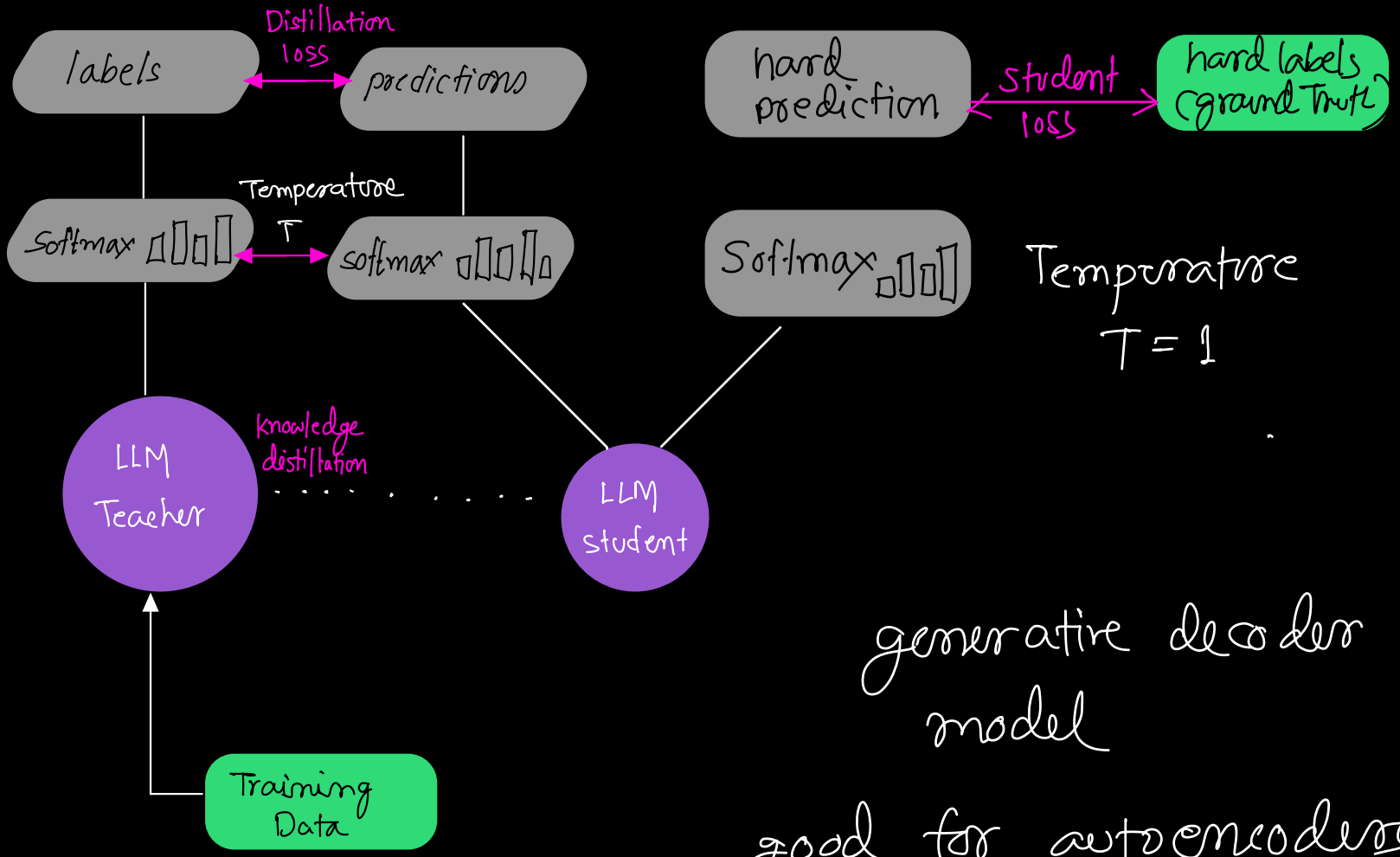
Quantization

Pruning





Train a smaller student model from a larger teacher model



generative decoder model

good for autoencoders like BERT

Post Training Quantization (PTQ)

Pruning

Remove model weights with values close to equal to zero

Rel

Models having difficulty

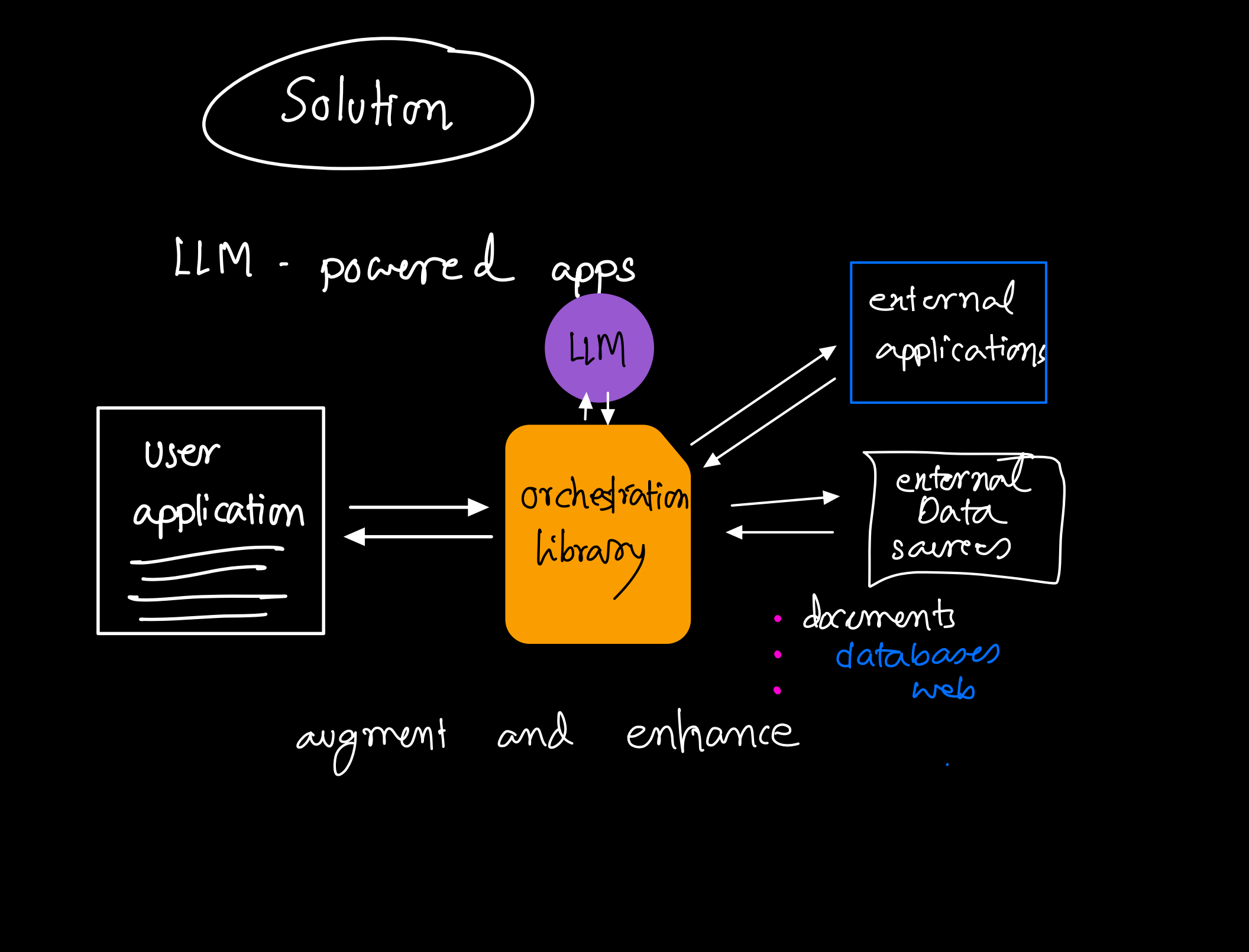
prompt → Model → Completion

Who is prime minister of UK ? → LLM → Boris Johnson — out of date

What is 40366/439 ? → LLM → 92.49 — wrong

What is a Martian Diameter ? → LLM → ? — hallucination

@ nlpwithindrajit

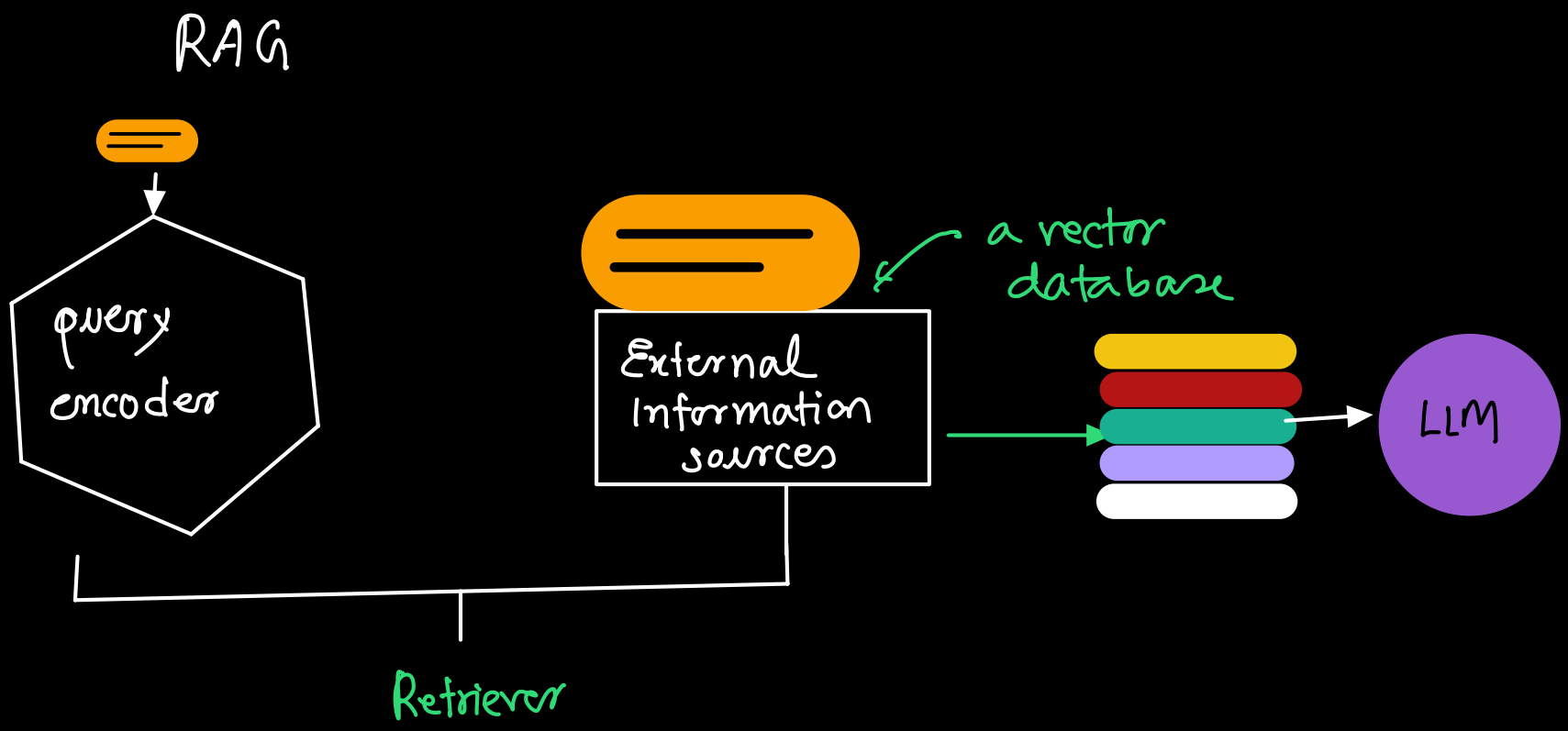

Solution
LLM - powered apps
LLM
external applications
User application
orchestration library
external Data sources
documents
databases
web
augment and enhance

Retrieval augmented generation (RAG)

- a framework to build LLM powered systems that make use of external data systems and applications.
- provide access to data that's not seen during training

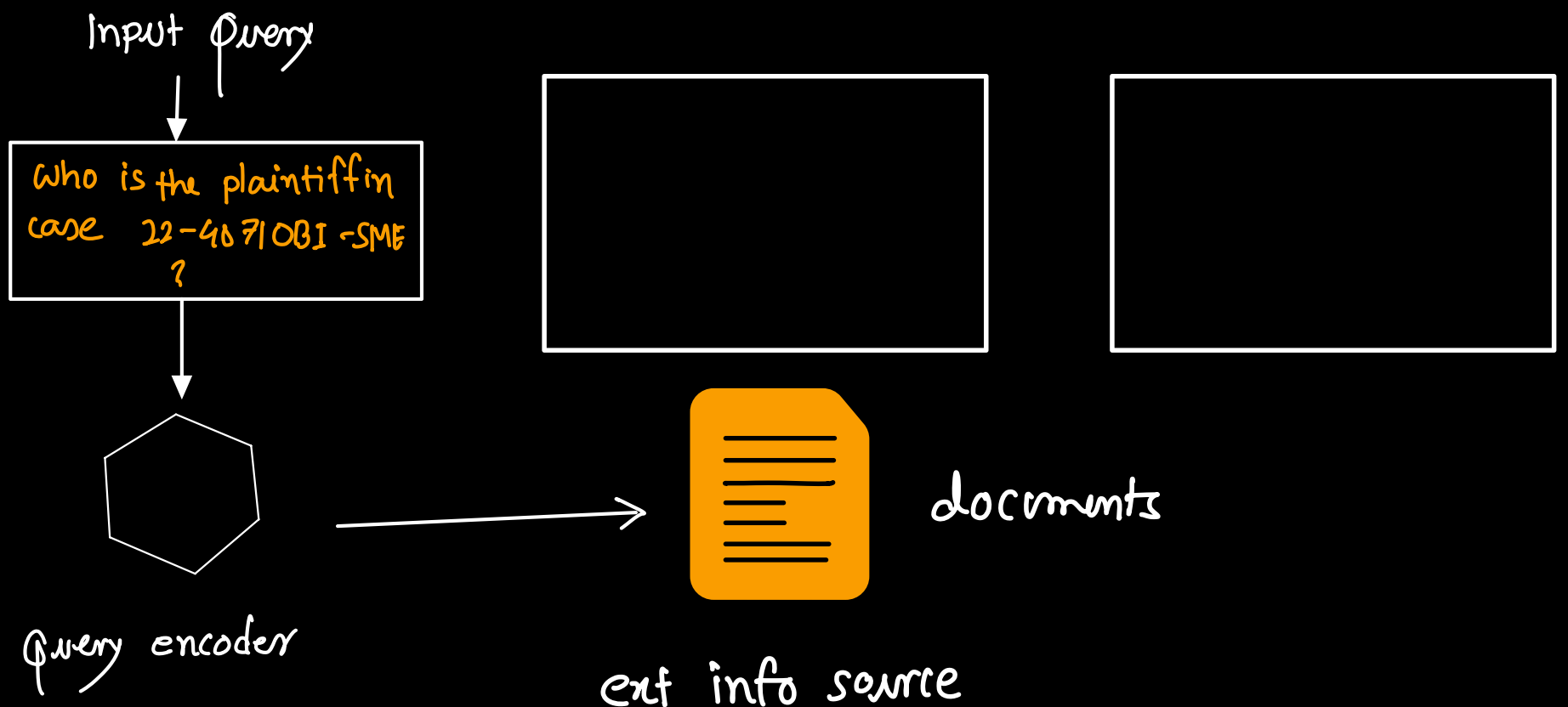
- one approach could be to give external/additional data during the inference time.
-

@nlpwithindrajit



@ nlpwith indrajit

Searching Legal documents

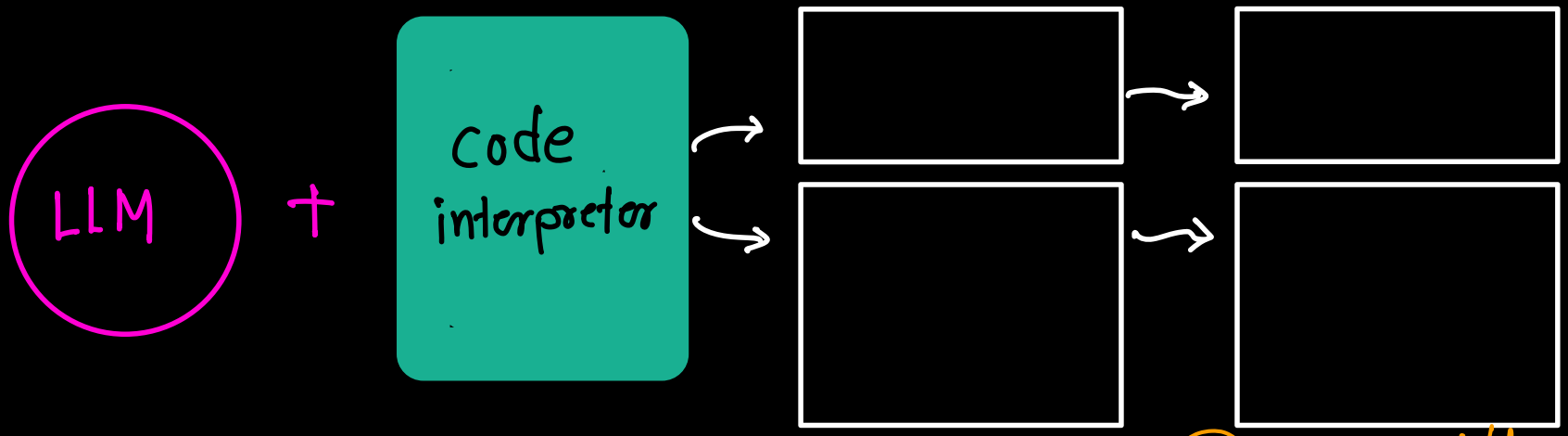


RAG Helps in reducing hallucination problems.

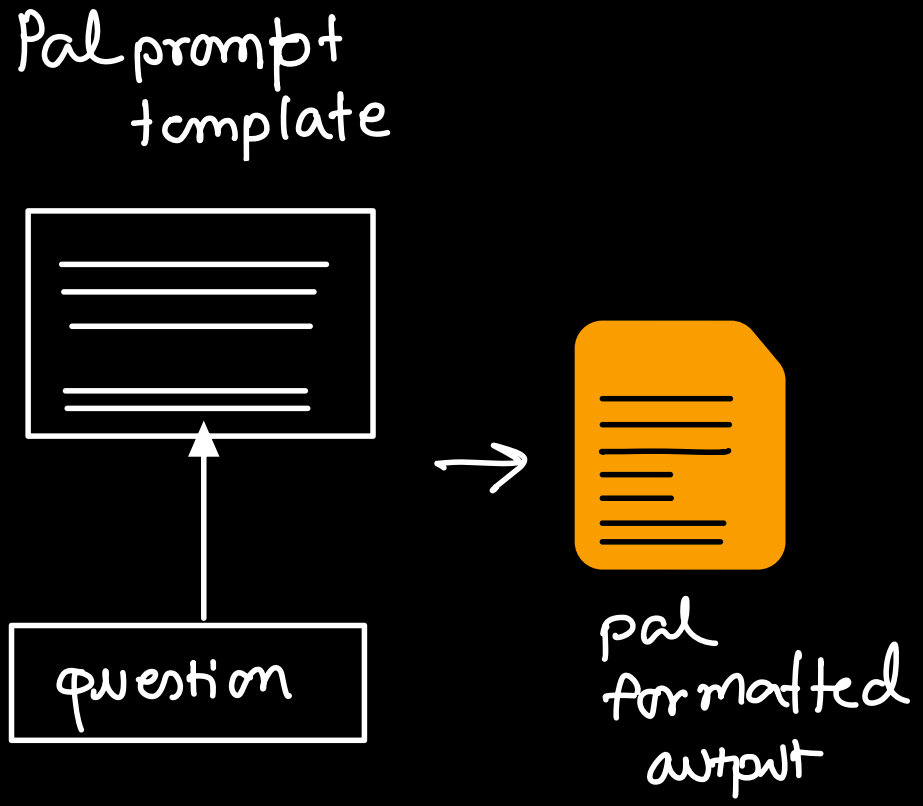
RAG takes small chunks of external data and store them & process them through LLMs to be stored on vector databases.

COT - Chain of thought

PAL - program - aided language models



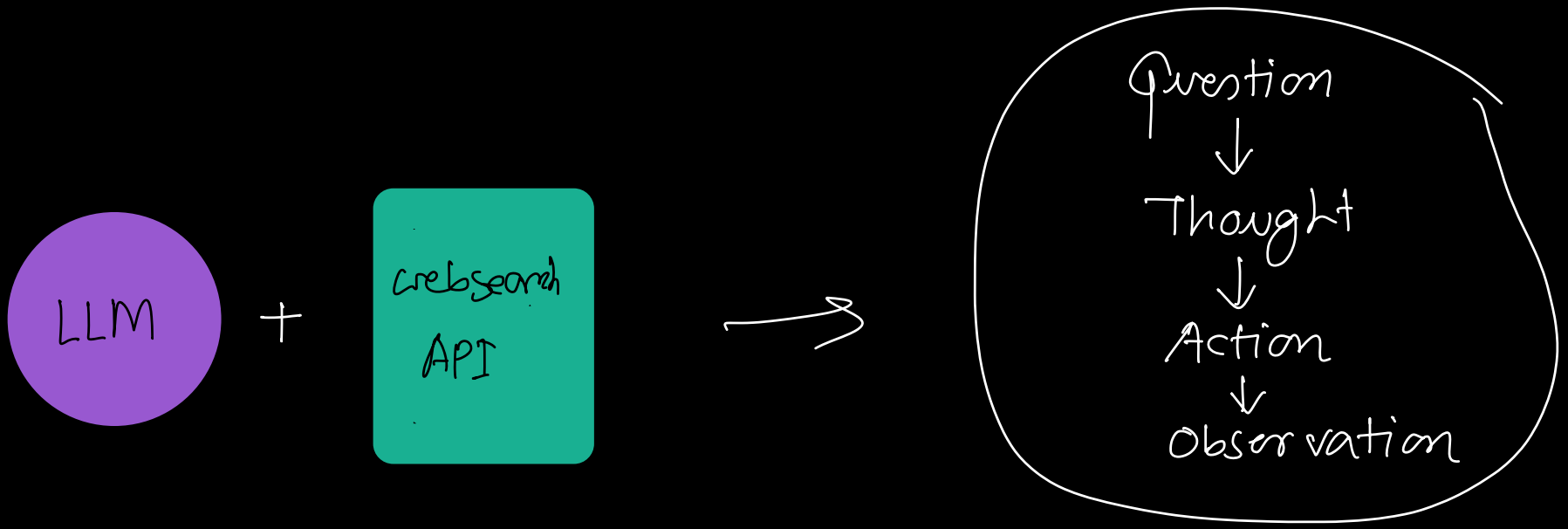
@nlpwithindrajit



@ nlpwith indrajit

ReAct : Combining reasoning and action

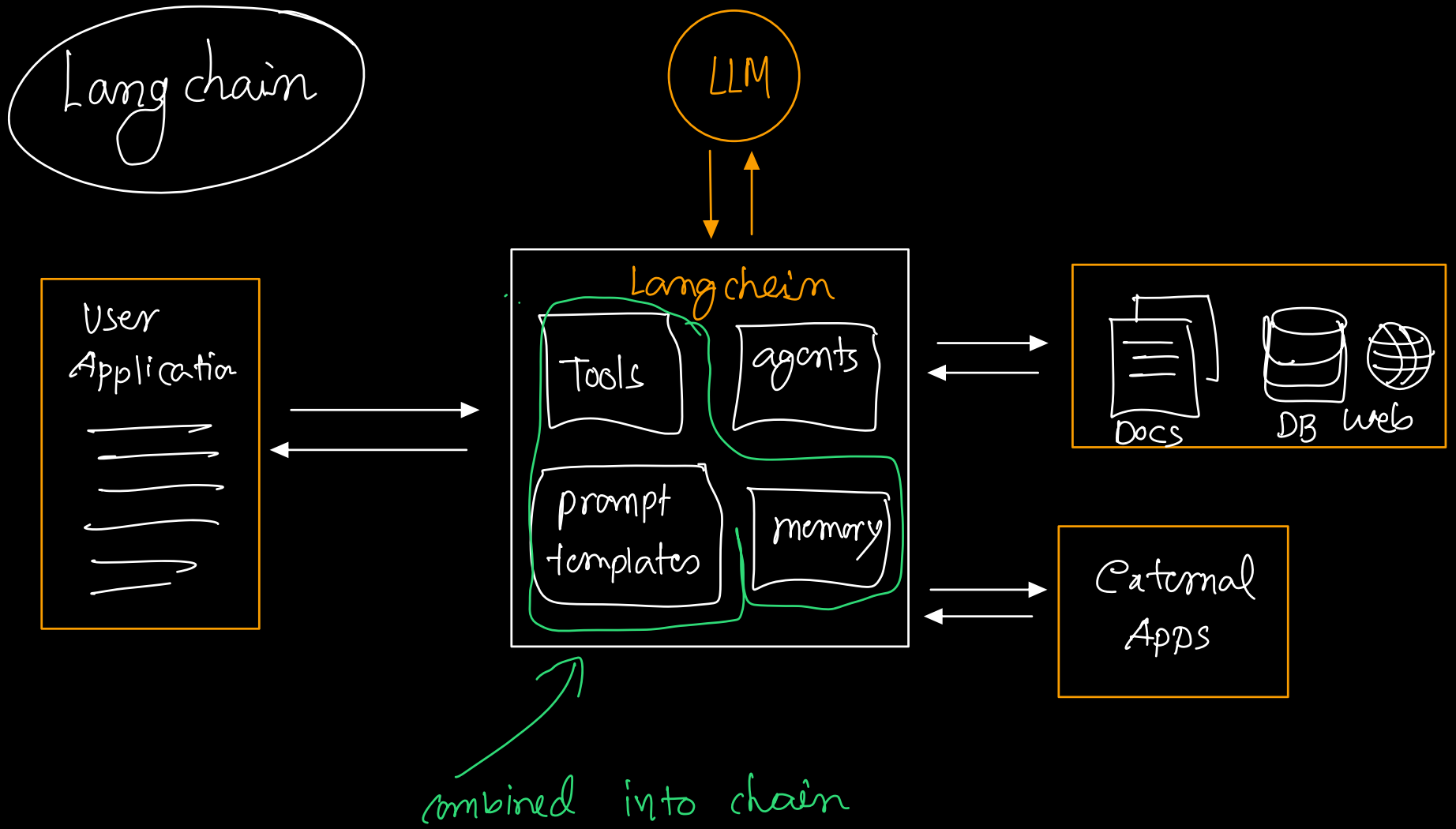
ReAct : Synergizing Reasoning and Action in LLMs



HotPot QA : multi step question answering

Fever :

@ nlpwithindrajit



@ nlpwith indrajit

@nlpwithindrajit

Thank You

inspired by AndrewNg's course

Generative Ai with Large Language Models. (coursera)