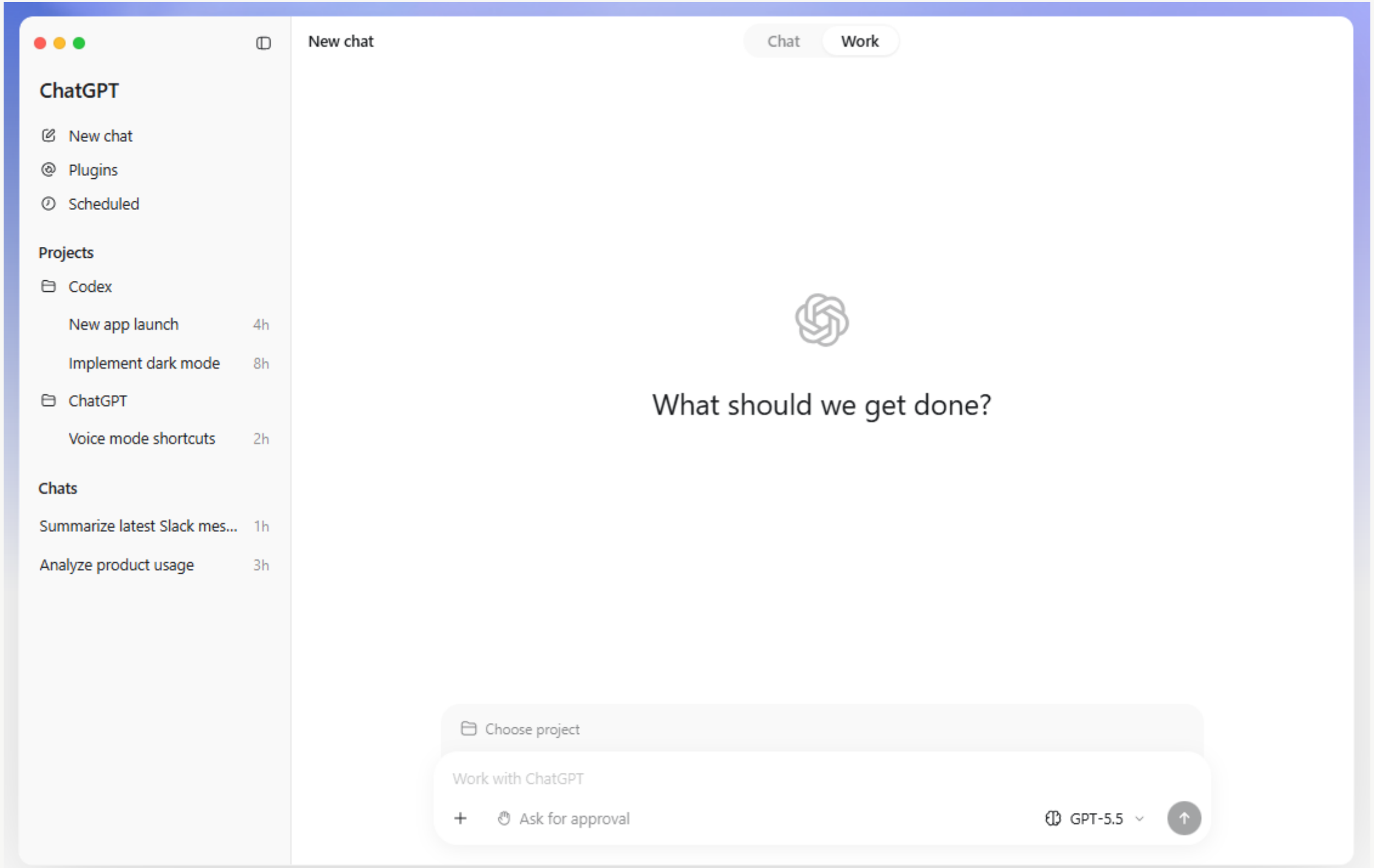
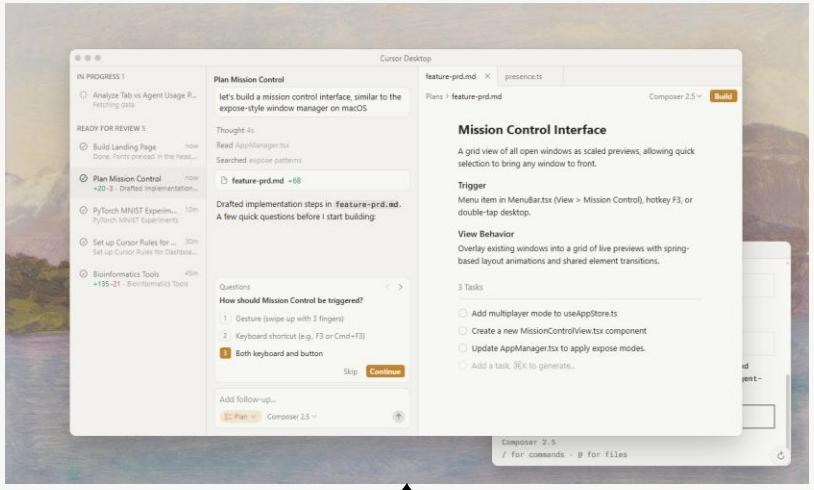
香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

CSC4801
AI-assisted Software Engineering

Lecture 01: The Principles of Large Language Models

Instructor: Jinsheng Ba | 2026 Fall


New chat
Chat
Work
ChatGPT
New chat
Plugins
Scheduled
Projects
Codex
New app launch 4h
Implement dark mode 8h
ChatGPT
Voice mode shortcuts 2h
Chats
Summarize latest Slack mes... 1h
Analyze product usage 3h
What should we get done?
Choose project
Work with ChatGPT
Ask for approval
GPT-5.5

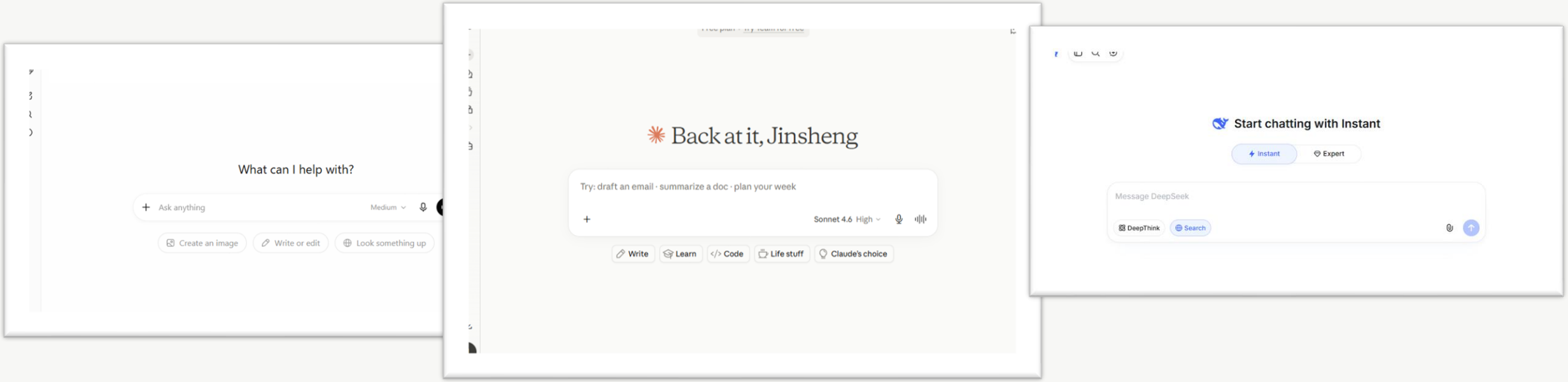


ChatGPT

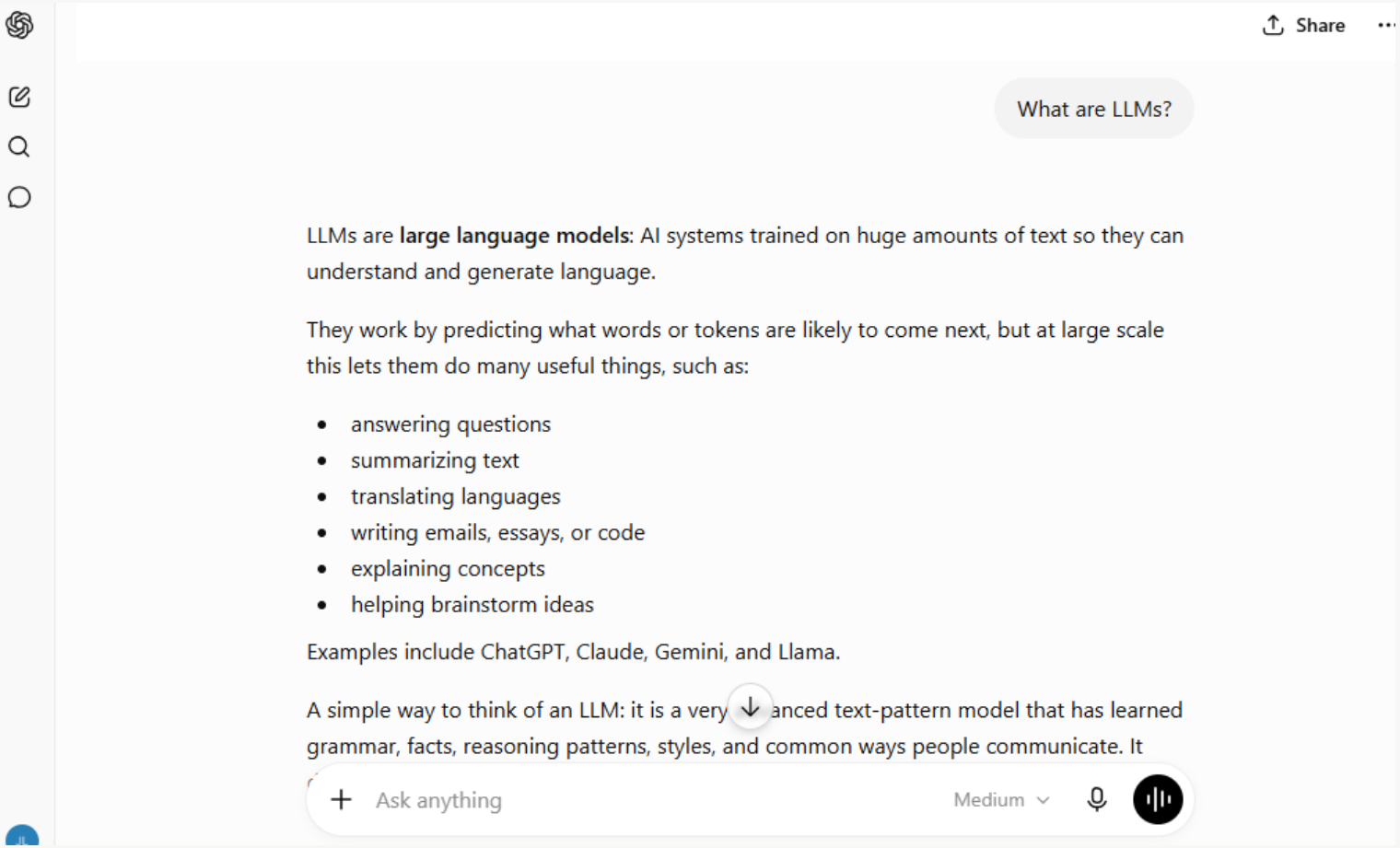
Claude

LLMs provide the core capabilities.

Goal



Understand the core principles of LLMs.

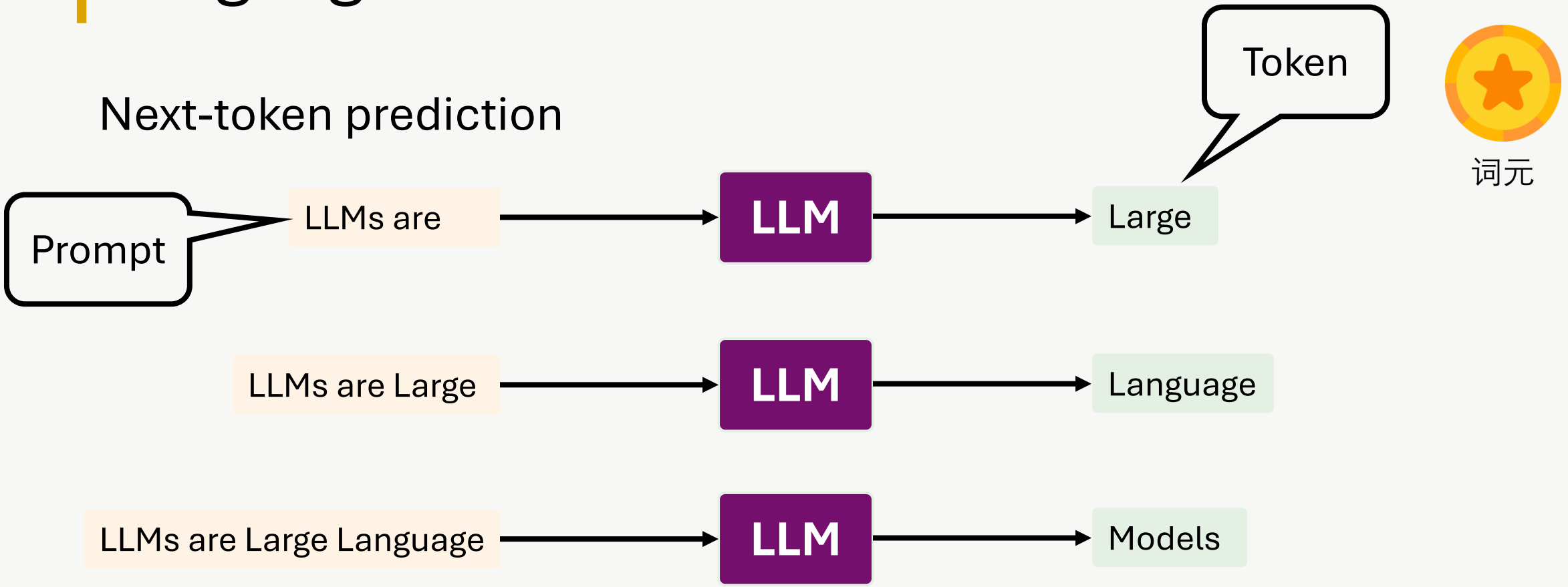


It is easy to use, but has super powerful and wide applications, especially in software engineering.

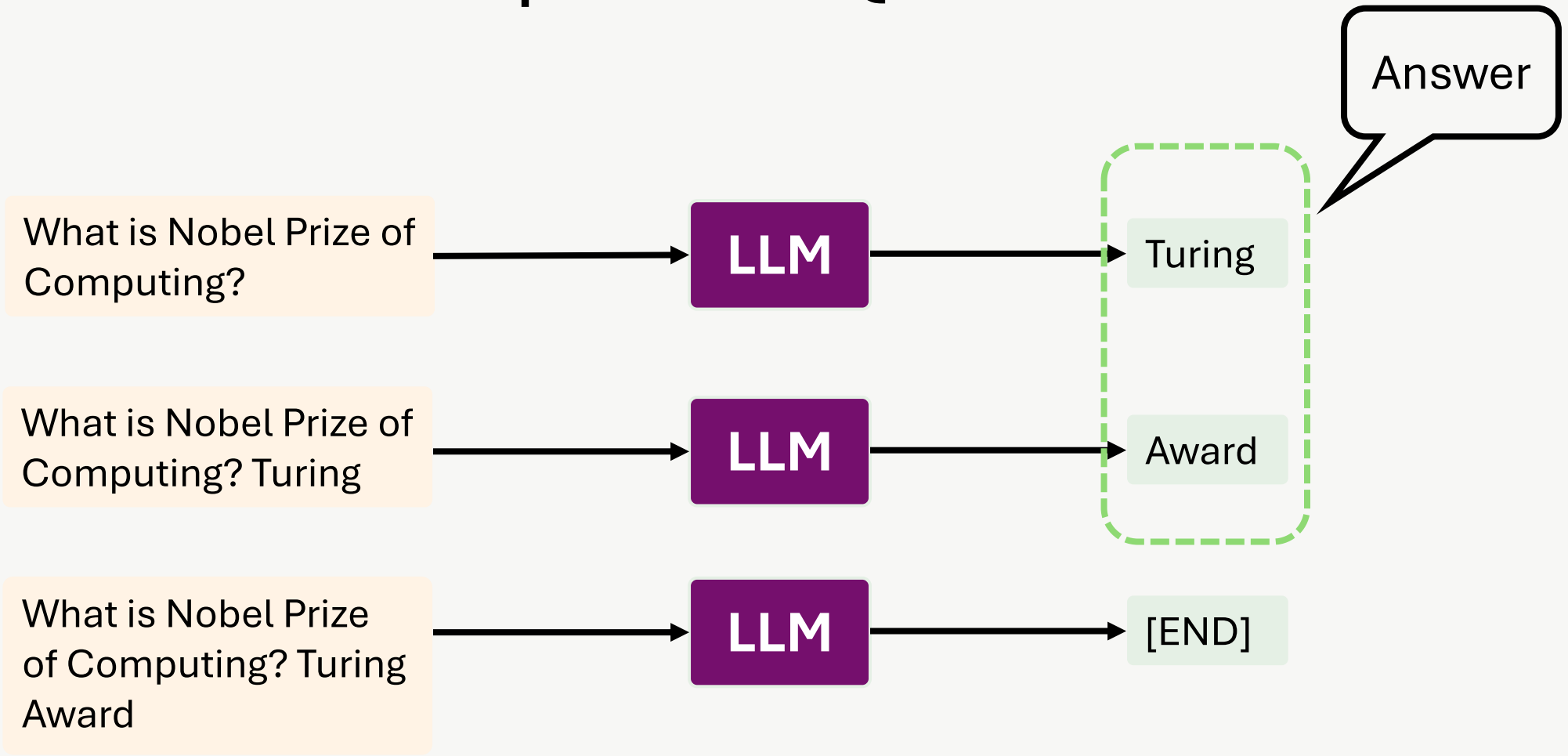
In your view, what defines a good model?

Language Models

Next-token prediction



How LLMs Response a Question?



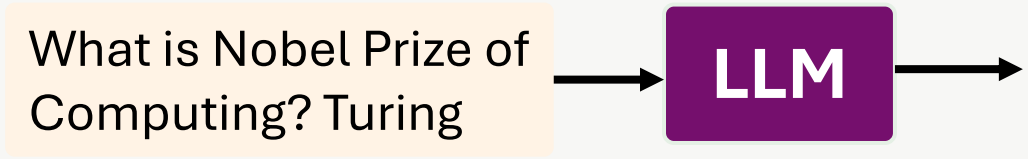
Can You Answer?

- Roses are red, violets are ___

- I opened the fridge and found a ___

How to Predict Next Token?

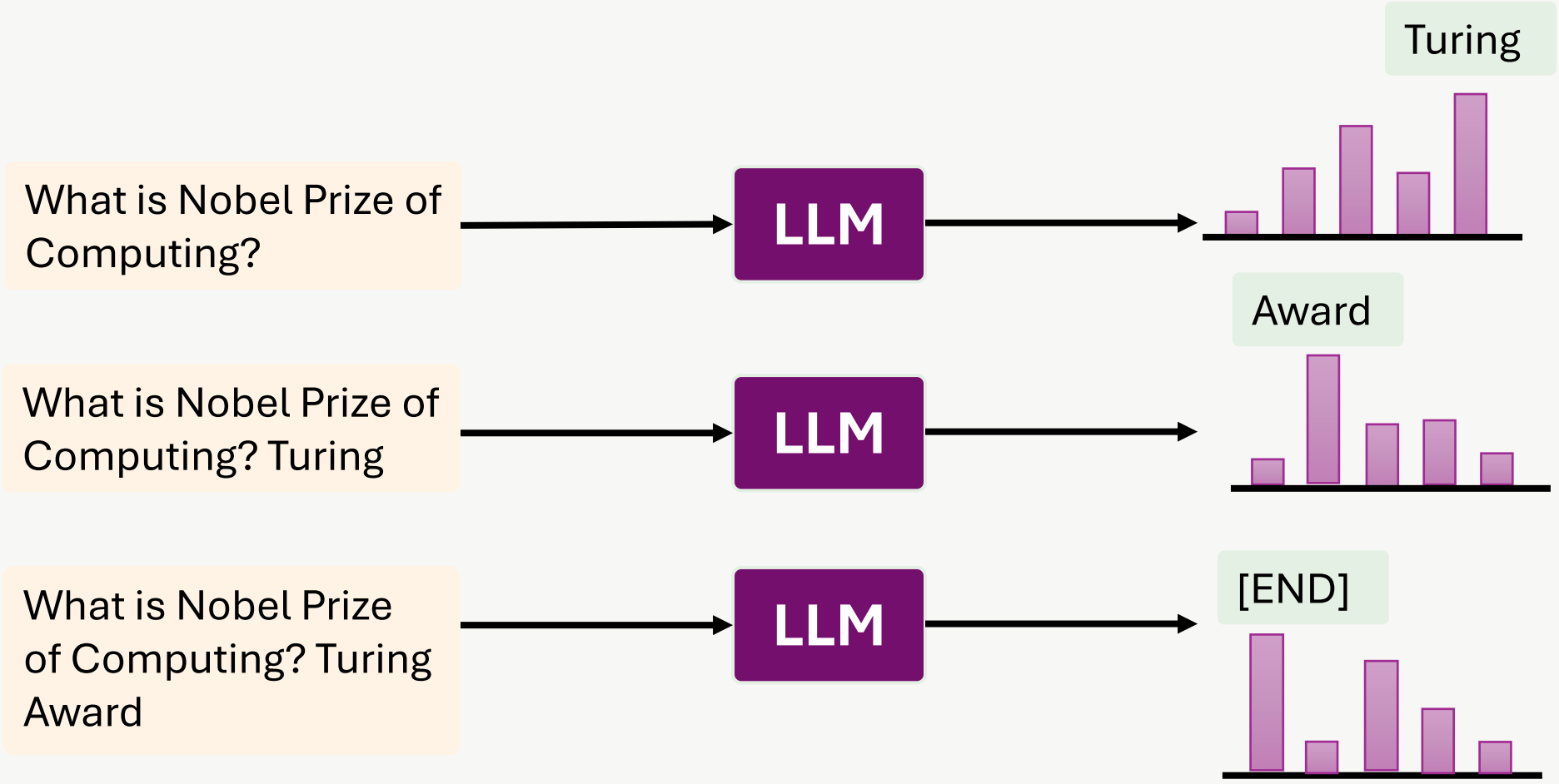
- A huge space for the next possible token
- How do LLMs choose the next one?



Token	Score
Award	0.5
Home	0.2
at	0.0000…
how	0.0000…
……	

Next-token possibility distribution

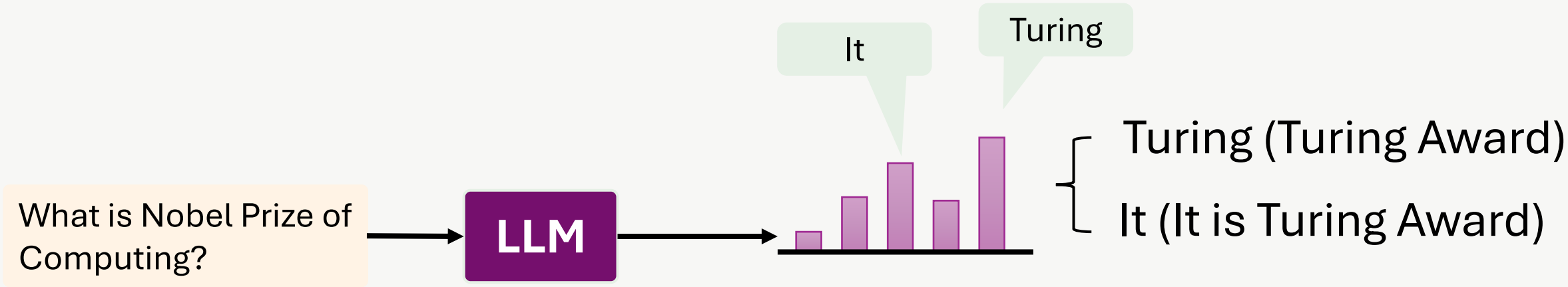
How LLMs Response a Question?



What is the problem of probabilistic?

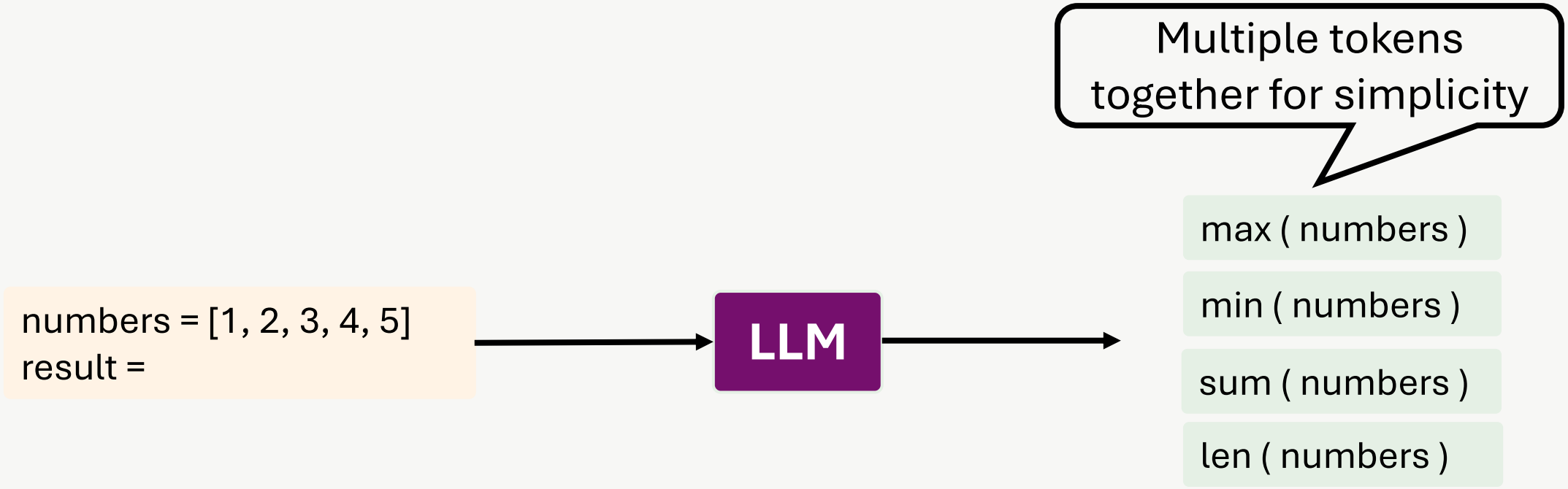
Probabilistic Output

- Probabilistic next-token prediction makes the output uncertain.



A better mode has a more accurate probability for the next-token prediction.

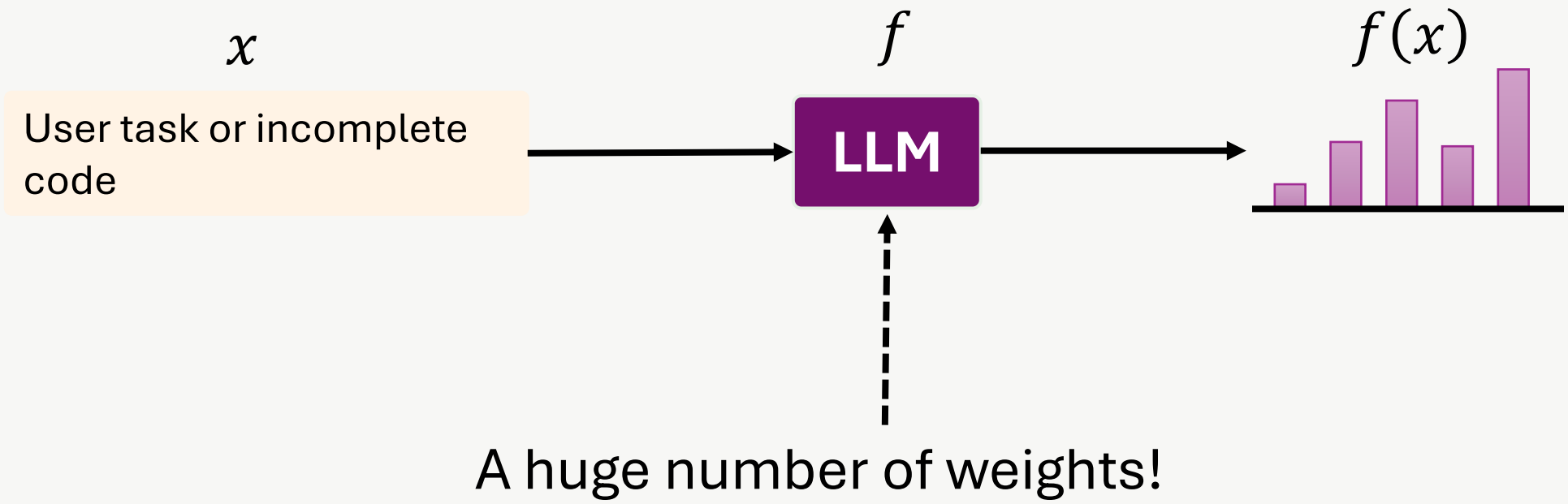
Next-Token Prediction is Challenging

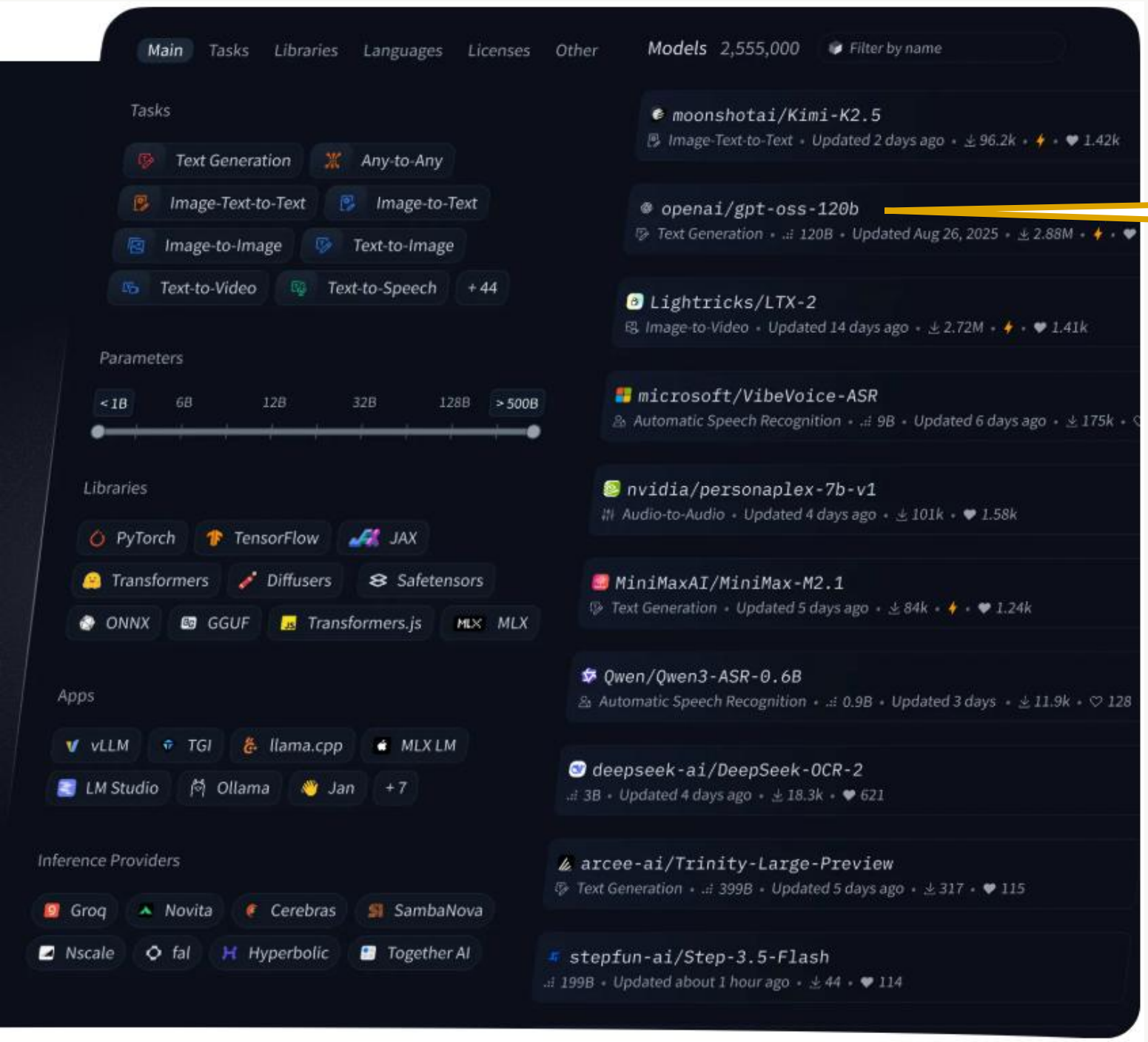


1. Understand the code intention
2. Correct function name
3. Correct variable name
4. Enclosed brackets

The Mechanisms Behind LLMs

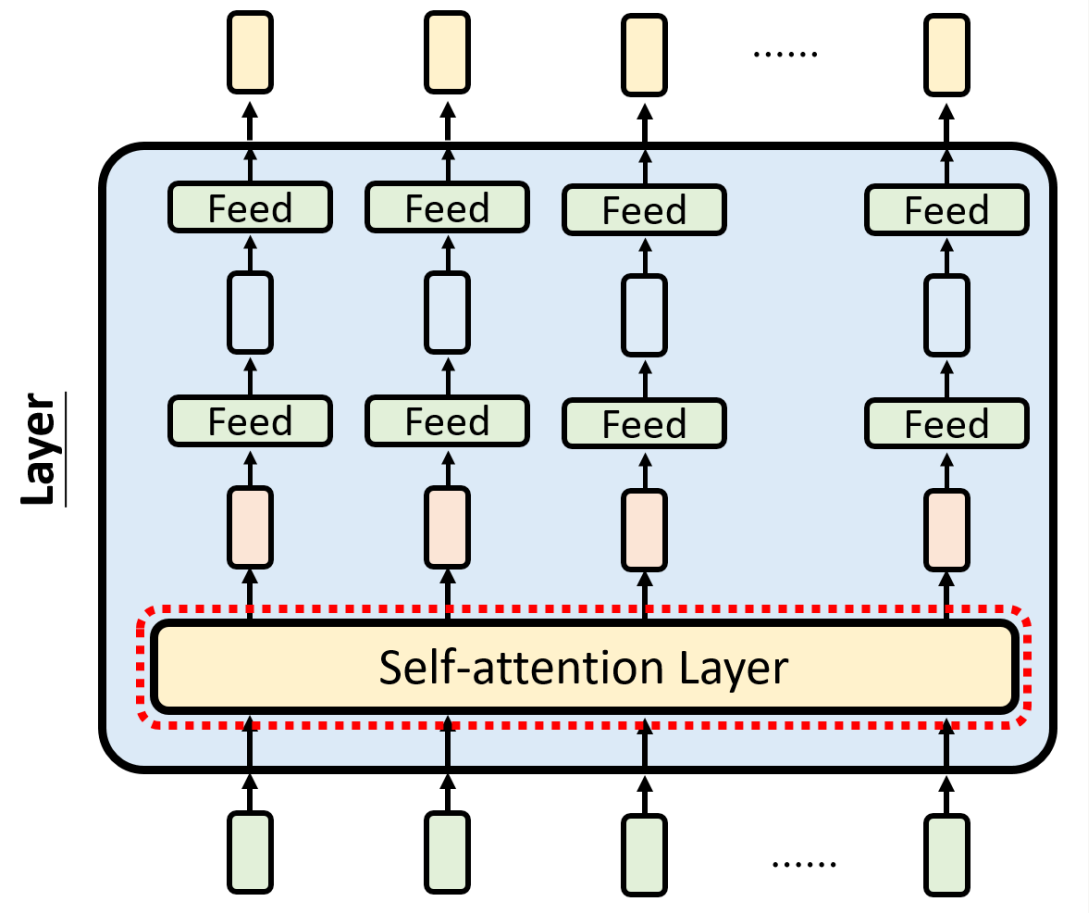
$e.g., f(x) = ax + b$





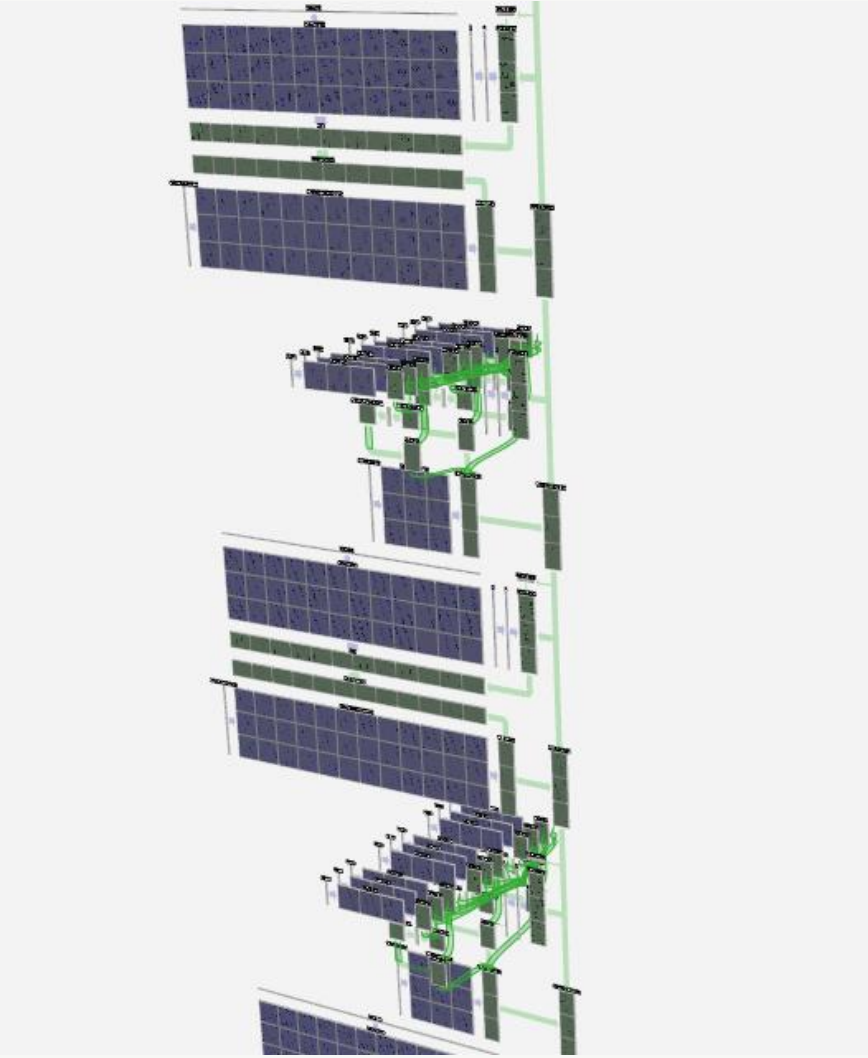
120 billions

Architecture of f: Transformers



- Reference material: https://jalammar.github.io/illustrated-transformer/
- We do not explain too detailed as it is not a LLM course

Architecture of f: Transformers



- https://bbycroft.net/llm

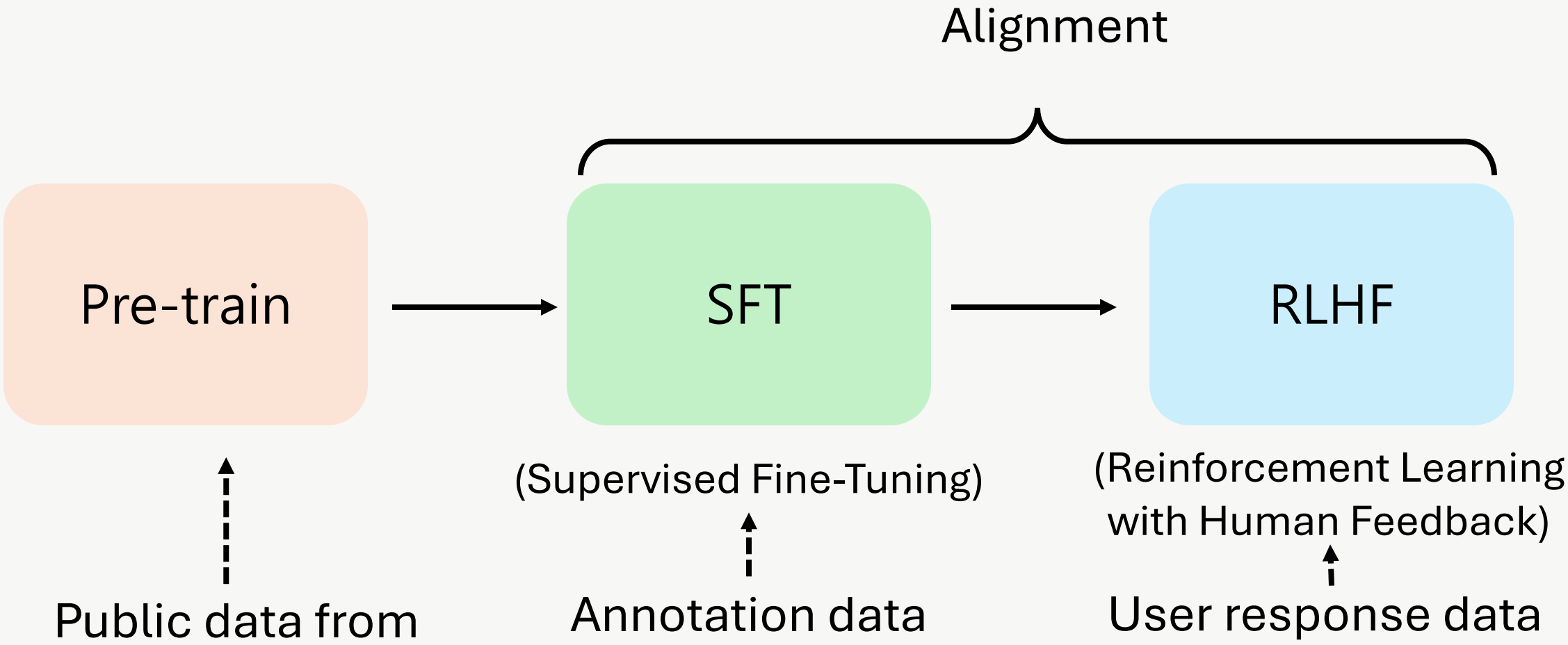
Where to Learn Weights?

- Public data from internet
  - GitHub: 420M+ projects/repositories
  - Stack overflow: biggest question-and-answer website for computer programmers.
- Annotation data
  - numbers = [1, 2, 3, 4, 5]
  - result = sum (numbers)
- User response data

max (numbers)

sum (numbers)

How to Learn Parameters?

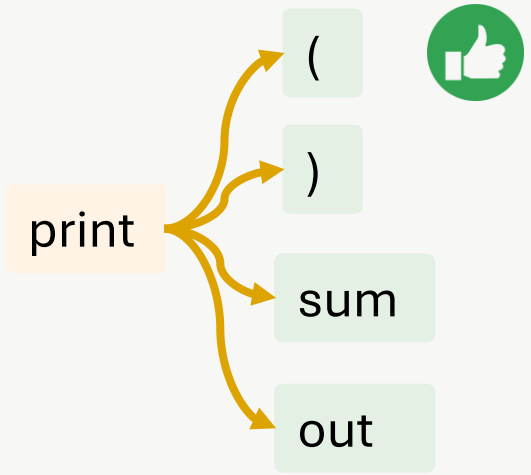


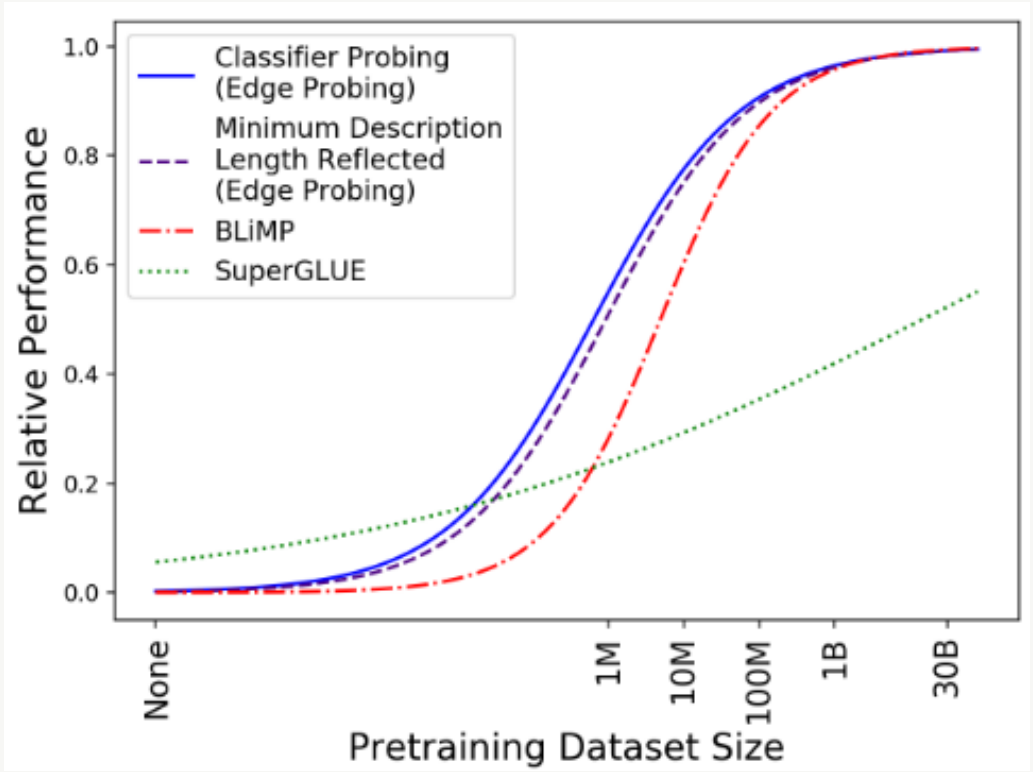
Learning Process

- Each stage's output is the input of the next stage

Pre-train

- Pretrain requires a huge amount of data





[1] Zhang, Y., Warstadt, A., Li, X., & Bowman, S. (2021, August). When do you need billions of words of pretraining data?. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing*.

Self-supervised Learning

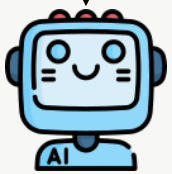
Internet

A huge amount of data

print(“hello world!”);

Input	Output
print	(
print(	“
print(“	hello
print(“hello	world
……	

Update weights



Few human effort involved.

How Large Data is Used?

LLaMA 3

https://arxiv.org/abs/2407.21783

- **Data.** Compared to prior versions of Llama (Touvron et al., 2023a,b), we improved both the quantity and quality of the data we use for pre-training and post-training. These improvements include the development of more careful pre-processing and curation pipelines for pre-training data and the development of more rigorous quality assurance and filtering approaches for post-training data. We pre-train Llama 3 on a corpus of about 15T multilingual tokens, compared to 1.8T tokens for Llama 2.

DeepSeek-V4

https://arxiv.org/pdf/2606.19348

and Heavily Compressed Attention (HCA) to improve long-context efficiency; (2) Manifold-Constrained Hyper-Connections (*m*HC) that enhance conventional residual connections; (3) and the Muon optimizer for faster convergence and greater training stability. We pre-train both models on more than 32T diverse and high-quality tokens, followed by a comprehensive post-training pipeline that unlocks and further enhances their capabilities. DeepSeek-V4-Pro-Max, the maximum reasoning effort mode of DeepSeek-V4-Pro, redefines the state-of-the-art for open models, outperforming its predecessors in core tasks. Meanwhile, DeepSeek-V4 series are highly efficient in long-context scenarios. In the one-million-token context setting, DeepSeek-V4-Pro requires only 27% of single-token inference FLOPs and 10% of KV cache compared with DeepSeek-V3.2. This enables us to routinely support one-million-token contexts, thereby

How Large Data is Used?

- Print out 15T of tokens.
- Assuming 100 sheets of paper are 1 cm thick
- The total thickness would be 1500 kilometers.

1000 tokens

1 Introduction

A central question in the discussion of large language models (LLMs) concerns the extent to which they *memorize* their training data versus how they *generalize* to new tasks and settings. Most practitioners seem to (at least informally) believe that LLMs do some degree of both: they *clearly* memorize parts of the training data—for example, are often able to reproduce large portions of training data verbatim [Carlini et al., 2023]—but they also seem to learn from this data, allowing them to generalize to new settings. The precise extent to which they do one or the other has massive implications for the practical and legal aspects of such models [Cooper et al., 2023]. Do LLMs truly produce new content, or do they only remix their training data? Should the act of training on copyrighted data be deemed unfair use of data, or should fair use be judged by the model's memorization? With respect to people, we distinguish plagiarising content from learning from it, but how should this extend to LLMs? The answer to such questions inherently relates to the extent to which LLMs memorize their training data.

However, even defining memorization for LLMs is challenging and many existing definitions leave a lot to be desired. Certain formulations claim that a passage from the training data is memorized if the LLM can reproduce it exactly [Nasr et al., 2023]. However, this ignores situations where, for instance, a prompt instructs the model to exactly repeat some phrase. Other formulations define memorization by whether or not prompting an LLM with a portion of text from the training set results in the completion of that training datum [Carlini et al., 2023]. But these formalisms rely fundamentally on the completions being a certain size, and typically very lengthy generations are required for sufficient certainty of memorization. More crucially, these definitions are too permissive because they ignore situations where model developers can (for legal compliance) post-hoc "align" an LLM by instructing their models not to produce certain copyrighted content [Ippolito et al., 2023]. But has such an instructed model really *not memorized* the sample in question, or does the model still contain all the information about the datum in its weights while it hides behind an illusion of compliance? Asking such questions becomes critical because this illusion of "unlearning" can often be easily broken as we show in Sections 4.1 and 4.3.

In this work, we propose a new definition of memorization based on a compression argument. Our definition posits that *a phrase present in the training data is memorized if we can make the model reproduce the phrase using a prompt (much) shorter than the phrase itself*. Operationalizing this definition requires finding the shortest adversarial input prompt that is specifically optimized to produce a target output. We call this ratio of input to output tokens the Adversarial Compression Ratio (ACR). In other words, memorization is inherently tied to whether a certain output can be represented in a *compressed* form, beyond what language models can do with typical text. We argue that such a definition provides an intuitive notion of memorization—if a certain phrase exists within the LLM training data (e.g., is not itself generated text) *and* it can be reproduced with fewer input tokens than output tokens, then the phrase must be stored somehow within the weights of the LLM. Although it may be more natural to consider compression in terms of the LLM-based notions of input/output perplexity, we argue that a simple compression ratio based on input/output token counts provides a more intuitive explanation to non-technical audiences, and has the potential to serve as a legal basis for important questions about memorization and permissible data use.

In addition to its intuitive nature, our definition has several other desirable qualities. We show that it appropriately ascribes many famous quotes as being memorized by existing LLMs (i.e. they have high ACR values). On the other hand, we find that text not in the training data of an LLM, such as samples posted on the internet after the training period, are not compressible, that is their ACR is low.

We examine several unlearning methods using ACR to show that they do not substantially affect the memorization of the model. That is, even after explicit finetuning, models asked to "forget" certain pieces of content are still able to reproduce them with a high ACR—in fact, not much smaller than with the original model. Our approach provides a simple and practical perspective on what memorization can mean, providing a useful tool for functional and legal analysis of LLMs.

2 Do We Really Need Another Notion of Memorization?

With LLMs ingesting more and more data, questions about their memorization are attracting attention [e.g. Carlini et al., 2019, 2021, Nasr et al., 2023, Zhang et al., 2023]. There remains a pressing need

2

How Large Data is Used?

- Print out 15T of tokens.
- Assuming 100 sheets of paper are 1 cm thick
- The total thickness would be 1500 kilometers.
- Suppose reading a page in 10 seconds, it requires 4756 years to complete.

1000 tokens

1 Introduction

A central question in the discussion of large language models (LLMs) concerns the extent to which they *memorize* their training data versus how they *generalize* to new tasks and settings. Most practitioners seem to (at least informally) believe that LLMs do some degree of both: they *clearly* memorize parts of the training data—for example, are often able to reproduce large portions of training data verbatim [Carlini et al., 2023]—but they also seem to learn from this data, allowing them to generalize to new settings. The precise extent to which they do one or the other has massive implications for the practical and legal aspects of such models [Cooper et al., 2023]. Do LLMs truly produce new content, or do they only remix their training data? Should the act of training on copyrighted data be deemed unfair use of data, or should fair use be judged by the model's memorization? With respect to people, we distinguish plagiarising content from learning from it, but how should this extend to LLMs? The answer to such questions inherently relates to the extent to which LLMs memorize their training data.

However, even defining memorization for LLMs is challenging and many existing definitions leave a lot to be desired. Certain formulations claim that a passage from the training data is memorized if the LLM can reproduce it exactly [Nasr et al., 2023]. However, this ignores situations where, for instance, a prompt instructs the model to exactly repeat some phrase. Other formulations define memorization by whether or not prompting an LLM with a portion of text from the training set results in the completion of that training datum [Carlini et al., 2023]. But these formalisms rely fundamentally on the completions being a certain size, and typically very lengthy generations are required for sufficient certainty of memorization. More crucially, these definitions are too permissive because they ignore situations where model developers can (for legal compliance) post-hoc "align" an LLM by instructing their models not to produce certain copyrighted content [Ippolito et al., 2023]. But has such an instructed model really *not memorized* the sample in question, or does the model still contain all the information about the datum in its weights while it hides behind an illusion of compliance? Asking such questions becomes critical because this illusion of "unlearning" can often be easily broken as we show in Sections 4.1 and 4.3.

In this work, we propose a new definition of memorization based on a compression argument. Our definition posits that *a phrase present in the training data is memorized if we can make the model reproduce the phrase using a prompt (much) shorter than the phrase itself*. Operationalizing this definition requires finding the shortest adversarial input prompt that is specifically optimized to produce a target output. We call this ratio of input to output tokens the Adversarial Compression Ratio (ACR). In other words, memorization is inherently tied to whether a certain output can be represented in a *compressed* form, beyond what language models can do with typical text. We argue that such a definition provides an intuitive notion of memorization—if a certain phrase exists within the LLM training data (e.g., is not itself generated text) *and* it can be reproduced with fewer input tokens than output tokens, then the phrase must be stored somehow within the weights of the LLM. Although it may be more natural to consider compression in terms of the LLM-based notions of input/output perplexity, we argue that a simple compression ratio based on input/output token counts provides a more intuitive explanation to non-technical audiences, and has the potential to serve as a legal basis for important questions about memorization and permissible data use.

In addition to its intuitive nature, our definition has several other desirable qualities. We show that it appropriately ascribes many famous quotes as being memorized by existing LLMs (i.e. they have high ACR values). On the other hand, we find that text not in the training data of an LLM, such as samples posted on the internet after the training period, are not compressible, that is their ACR is low.

We examine several unlearning methods using ACR to show that they do not substantially affect the memorization of the model. That is, even after explicit finetuning, models asked to "forget" certain pieces of content are still able to reproduce them with a high ACR—in fact, not much smaller than with the original model. Our approach provides a simple and practical perspective on what memorization can mean, providing a useful tool for functional and legal analysis of LLMs.

2 Do We Really Need Another Notion of Memorization?

With LLMs ingesting more and more data, questions about their memorization are attracting attention [e.g. Carlini et al., 2019, 2021, Nasr et al., 2023, Zhang et al., 2023]. There remains a pressing need

2

What happens if data is exhausted?

We lost our uniqueness

I would **not** frame the paper as:

> "Use AI to generate code."

That sounds incremental.

Instead:

> **We control AI to generate repeated code under bounded scope**

Not....., instead...


Not... Instead . Instead . Not...
Not.... Not.... Instead . Not...
Not...
Not.
Not
Instead
Not
Instead

Scaling Law

- Within limited pretraining budget, how should we allocate the budget to achieve the best possible model?
- More training data?
- More model parameters?
- Both?
- Model architectures?

Training = Spending Compute

- An approximation for transformer language models:

$$C \approx 6ND$$

N: number of model parameters
D: number of tokens
C: compute; floating point operations (FLOPs)

Scaling Law: https://arxiv.org/abs/2001.08361

Training = Spending Compute

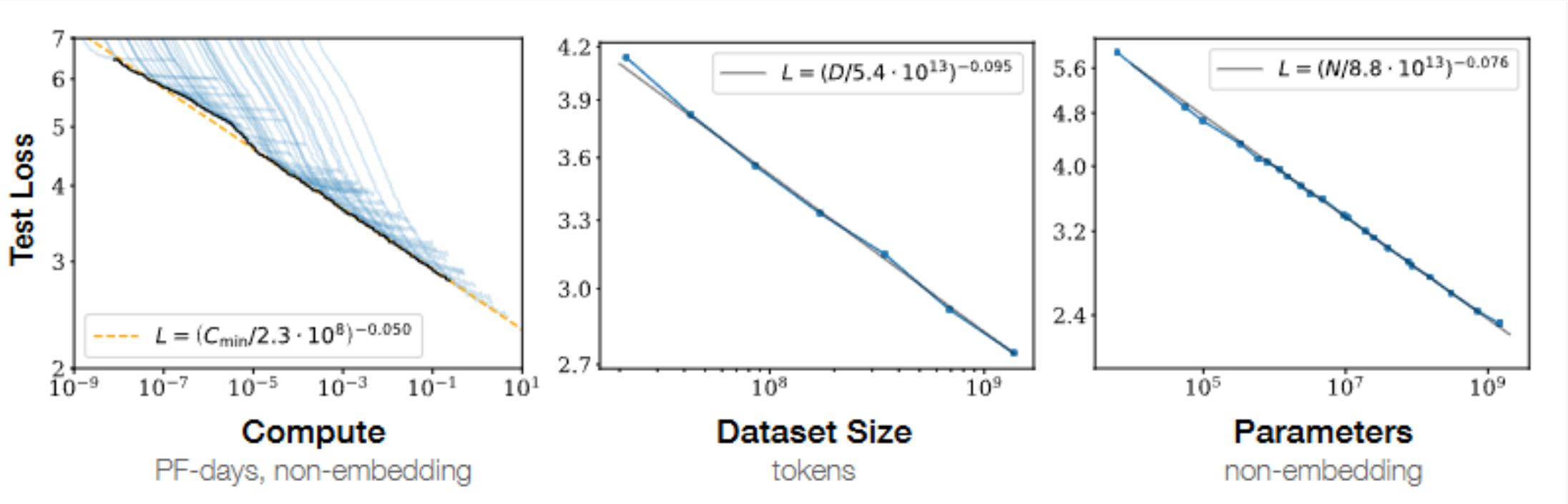
- For example, Llama 3:

$$C \approx 6 \times 70 \text{ billion} \times 15 \text{ trillion}$$
$$= 6.3 \times 10^{24} \text{ FLOPs}$$

N: number of model parameters
D: number of tokens
C: compute; floating point operations (FLOPs)

Training = Spending Compute

- **Data scaling:** change D, what happens to loss?
- **Model scaling:** change N, what happens to loss?
- **Compute scaling:** should we increase compute by increasing D or increasing N?

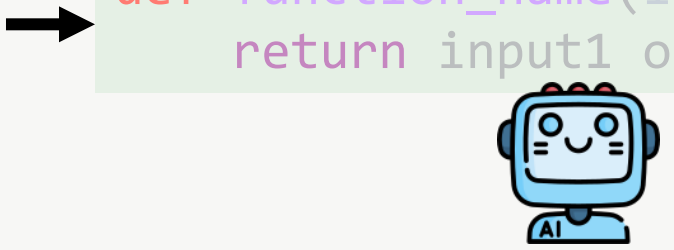


https://arxiv.org/pdf/2001.08361

Compression

- LLMs are not remembering everything from the training.
- LLMs compress all knowledge they have seen.

```
1. def add(a, b):
    return a + b
2. def multiply(a, b):
    return a * b
3. a / b
```

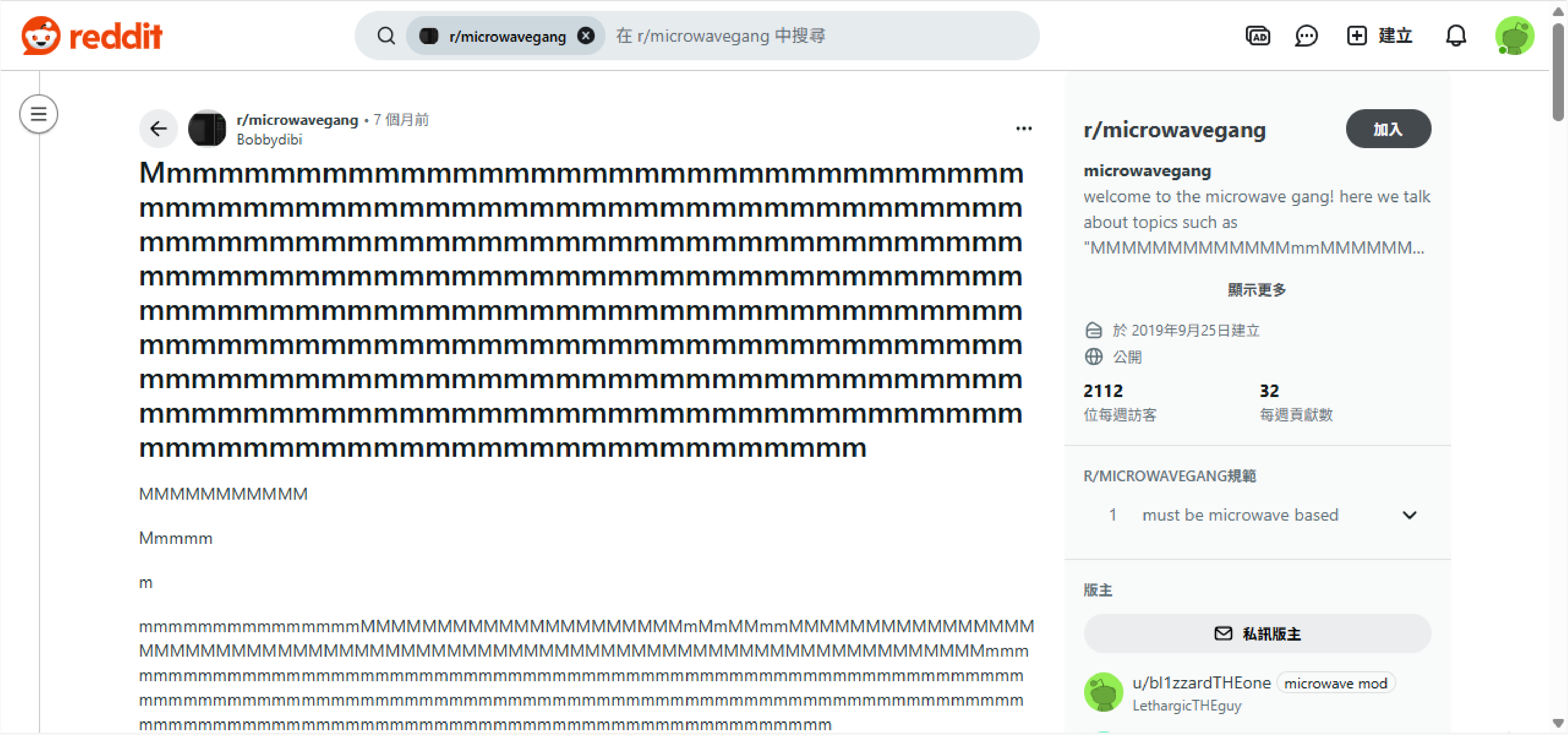


```
def function_name(input1,input2):
    return input1 operator input2
```

```
def divide(a, b):
```

```
return a / b
```

Is More Data Better?

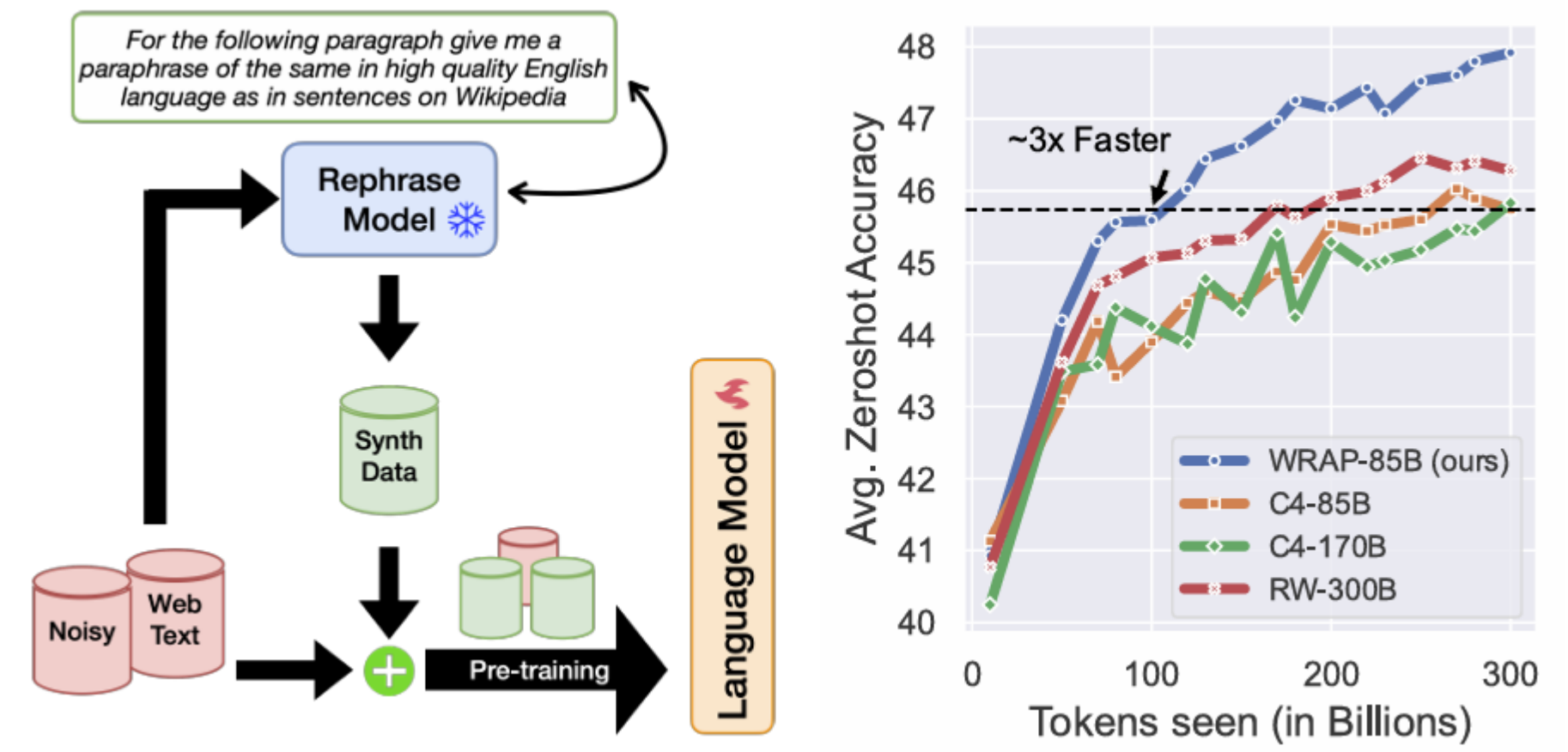


Quality is also important

Date Cleaning

Rephrasing the Web

https://arxiv.org/abs/2401.16380



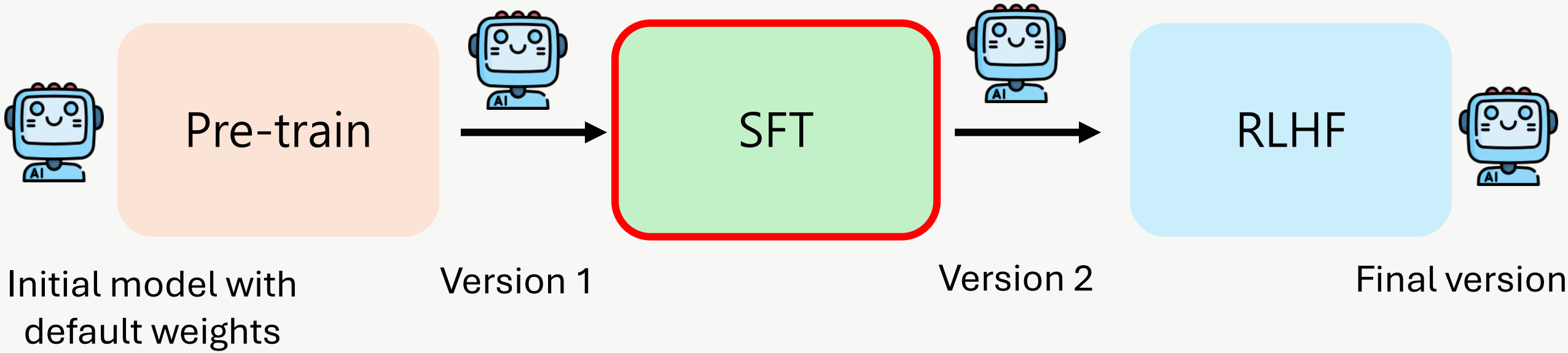
Pre-train

- Pre-trained models have the chance to make the correct answer
- The chance may not be good enough.
- The pre-trained model is like a rough gem; it still needs meticulous polishing.
- Next step: SFT & RL increase the chances for the correct answers.

Reasoning with Sampling: Your Base Model is Smarter Than You Think. https://arxiv.org/abs/2510.14901

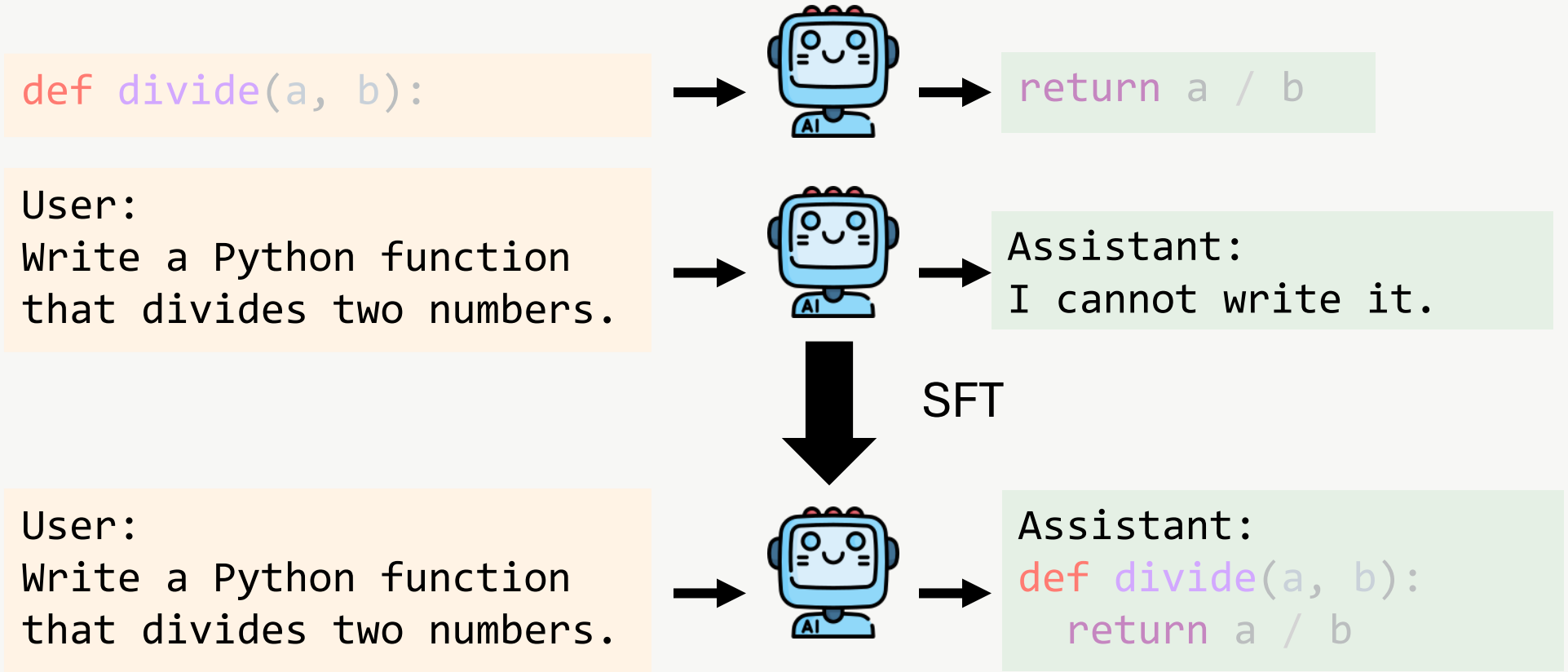
SFT

- Each stage's output is the input of the next stage



SFT

- SFT teaches the model: “How should I answer a user?”



SFT

- Use few but high-quality data to further adjust weights.
- LLaMA2

https://arxiv.org/abs/2307.09288

Quality Is All You Need. Third-party SFT data is available from many different sources, but we found that many of these have insufficient diversity and quality — in particular for aligning LLMs towards dialogue-style instructions. As a result, we focused first on collecting several thousand examples of high-quality SFT data, as illustrated in Table 5. By setting aside millions of examples from third-party datasets and using fewer but higher-quality examples from our own vendor-based annotation efforts, our results notably improved. These findings are similar in spirit to Zhou et al. (2023), which also finds that a limited set of clean instruction-tuning data can be sufficient to reach a high level of quality. We found that SFT annotations in the order of tens of thousands was enough to achieve a high-quality result. We stopped annotating SFT after collecting a total of 27,540 annotations. Note that we do not include any Meta user data.

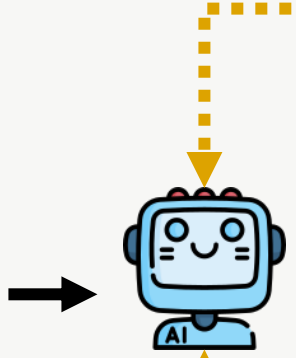
RLHF

- Each stage's output is the input of the next stage

RLHF

- Reinforcement Learning with Human Feedback
- RLHF teaches the model: “Which answer do humans prefer?”

```
User:
Write a Python function
that divides two numbers.
```




```
Assistant:
def divide(a, b):
  return a / b
```

```
Assistant:
def divide(a, b):
    if b == 0:
        return "Error"
    return a / b
```

No new training data!
Use preferences as feedback
to adjust weights.

Which One Do You Prefer?

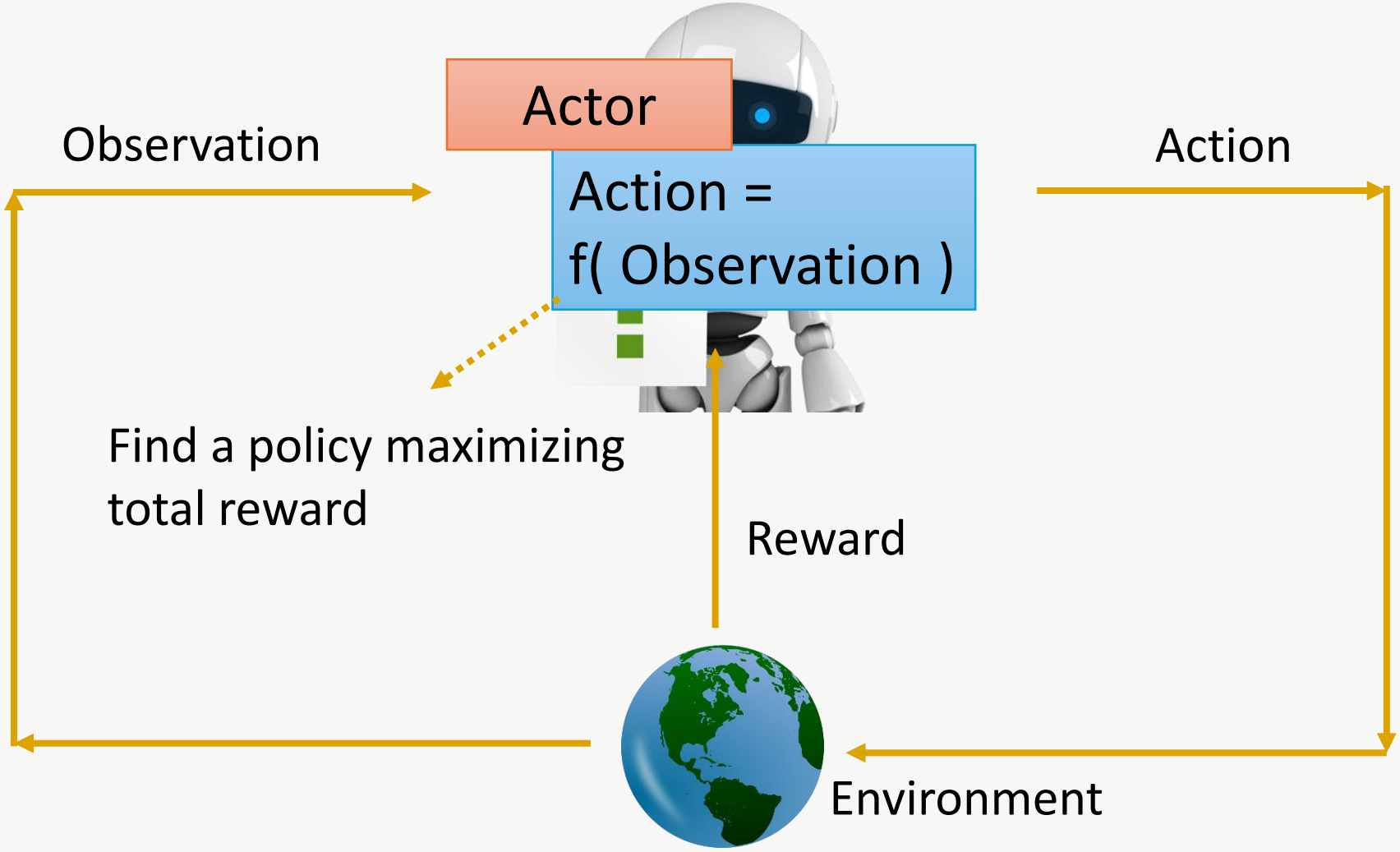
> User: "I'm only starting to study the day before the exam—is there any hope?"

A:
> "Of course! You can definitely get a high score—go for it!"

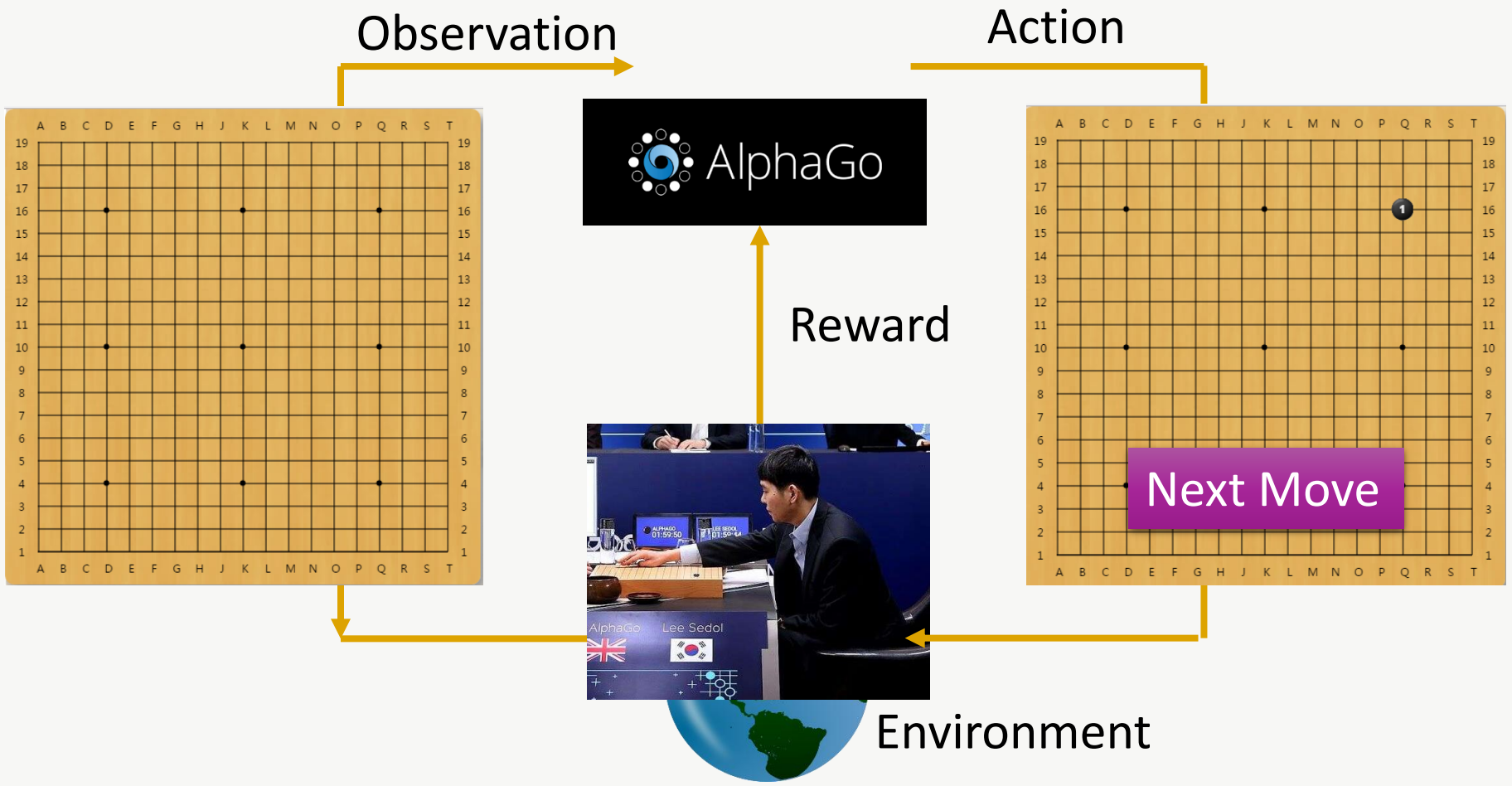
B:
> "It's impossible to predict the outcome right now. Tell me the scope of the exam, your current level of understanding, and how much time you have available, and I can help you set priorities."

C:
> "One day is generally not enough to systematically master an entire course."

Reinforcement Learning

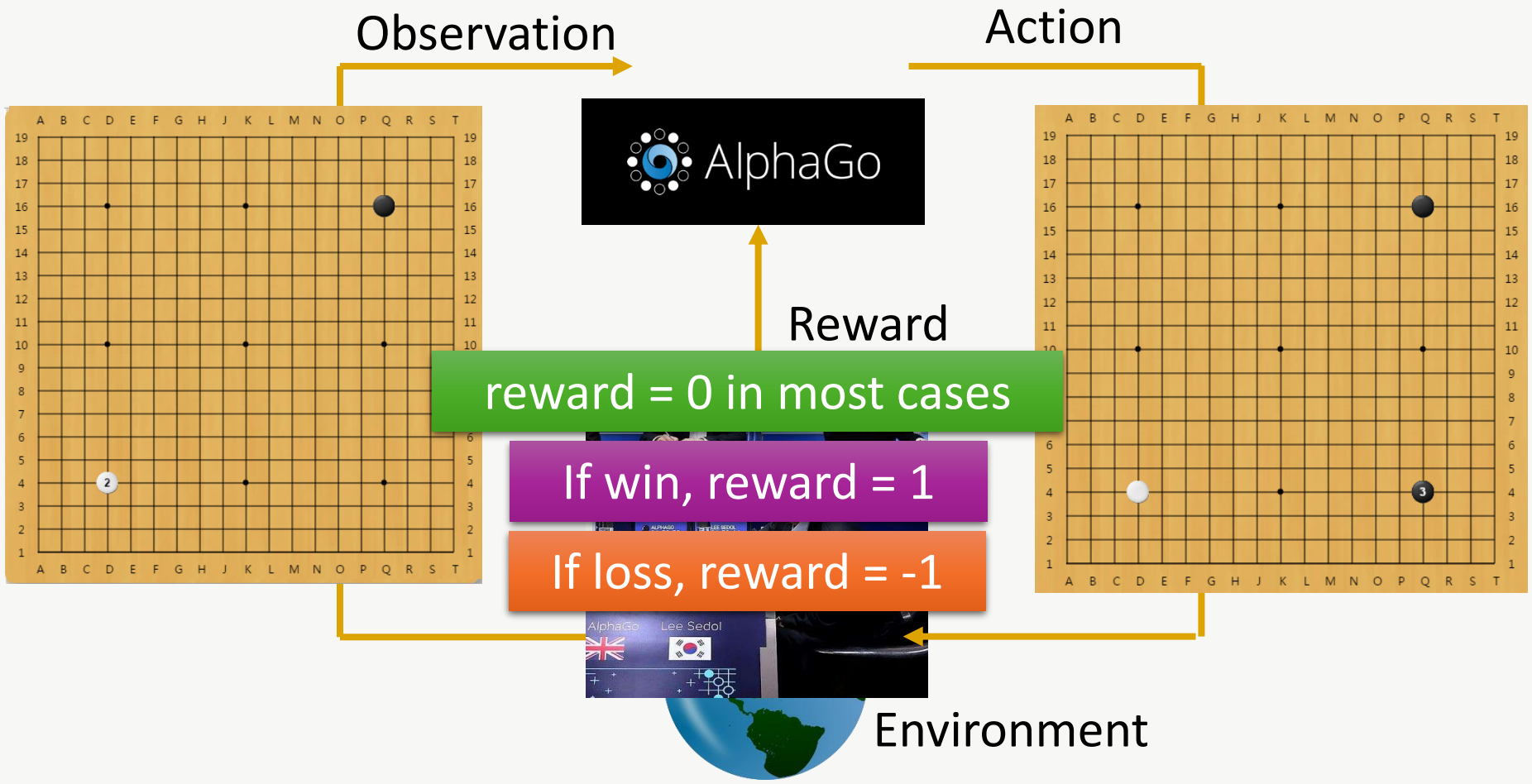


Reinforcement Learning

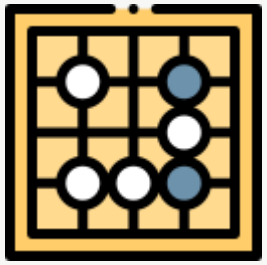


Reinforcement Learning

Find an actor maximizing expected reward.



Reinforcement Learning



Chessboard

AlphaGo

“5-5”

The next move

Prompt

LLM

The next token

The Reward is Important

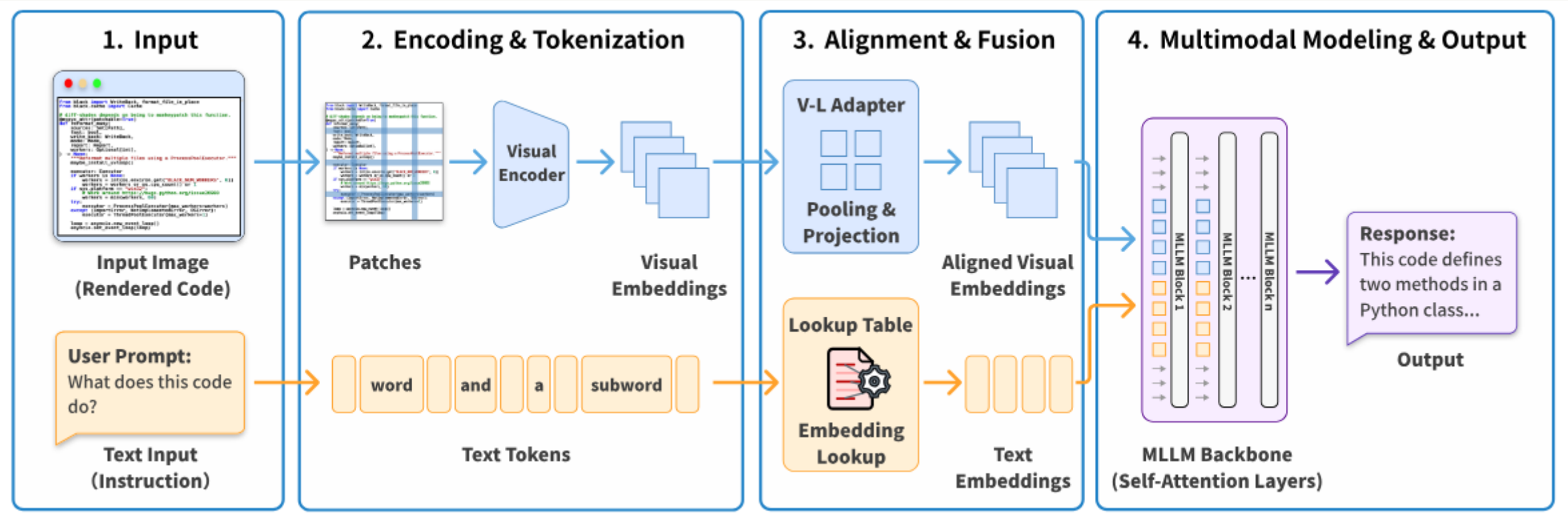
https://openai.com/index/faulty-reward-functions/?video=745142691

Learning Process

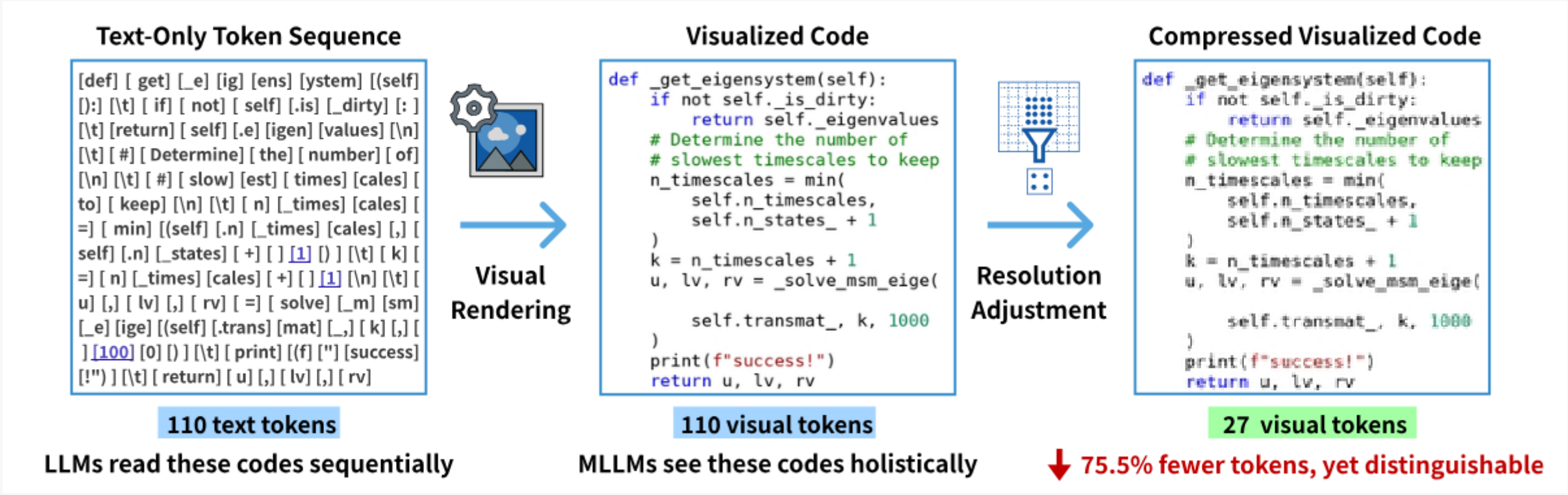
- Each stage's output is the input of the next stage

Are tokens must be text?

Image Representations of Code



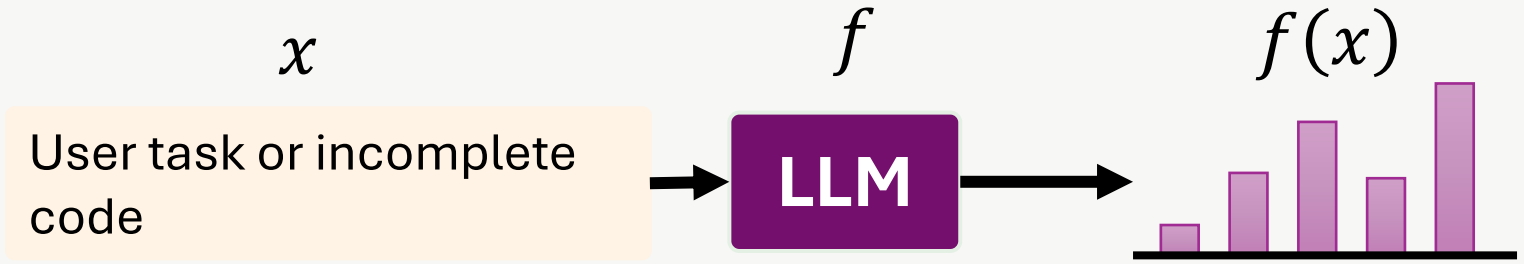
https://arxiv.org/pdf/2602.01785



https://arxiv.org/pdf/2602.01785

Black-box LLMs

- Typically commercial
- Huge
- Powerful
- We do not know how f works and is produced



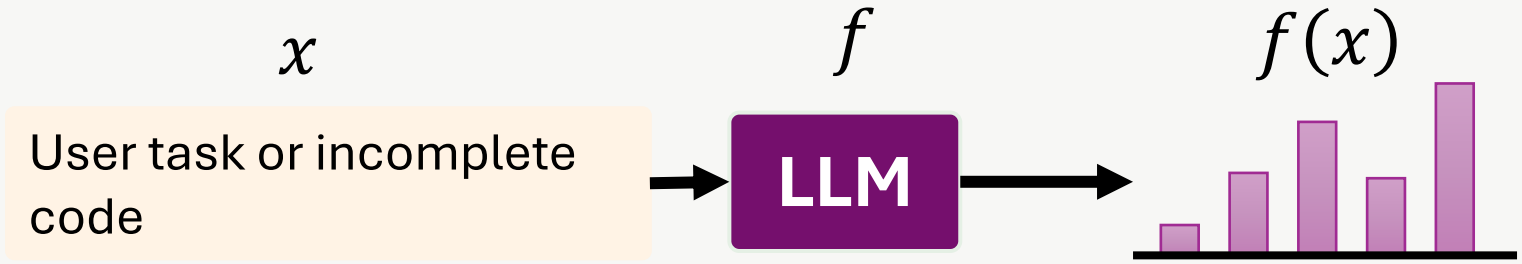
Gemini

ChatGPT

Claude

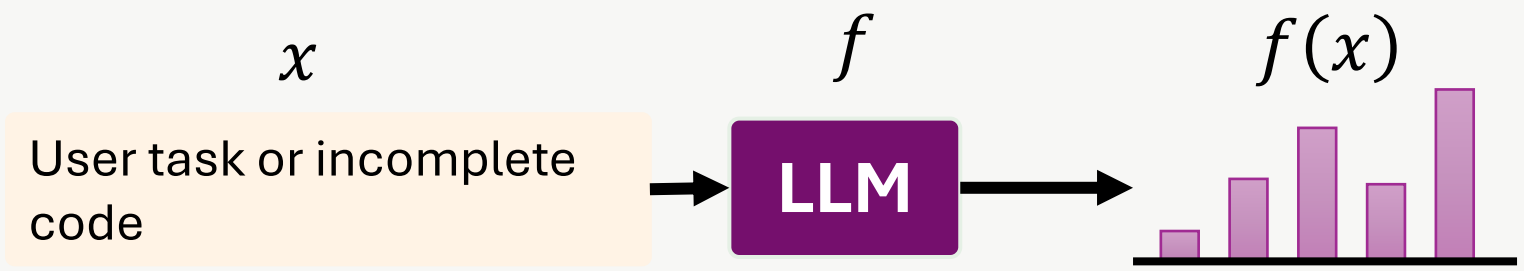
Open-source LLMs

- You can see the data, data filtering, tokenizer, training code, hyperparameters, checkpoints, logs, and evaluation process.
- We know how f is produced



Open-weight LLMs

- A trade-off between black box and open source
- Only parameters are public
- We know the internal of f
- We can download and deploy on local machines



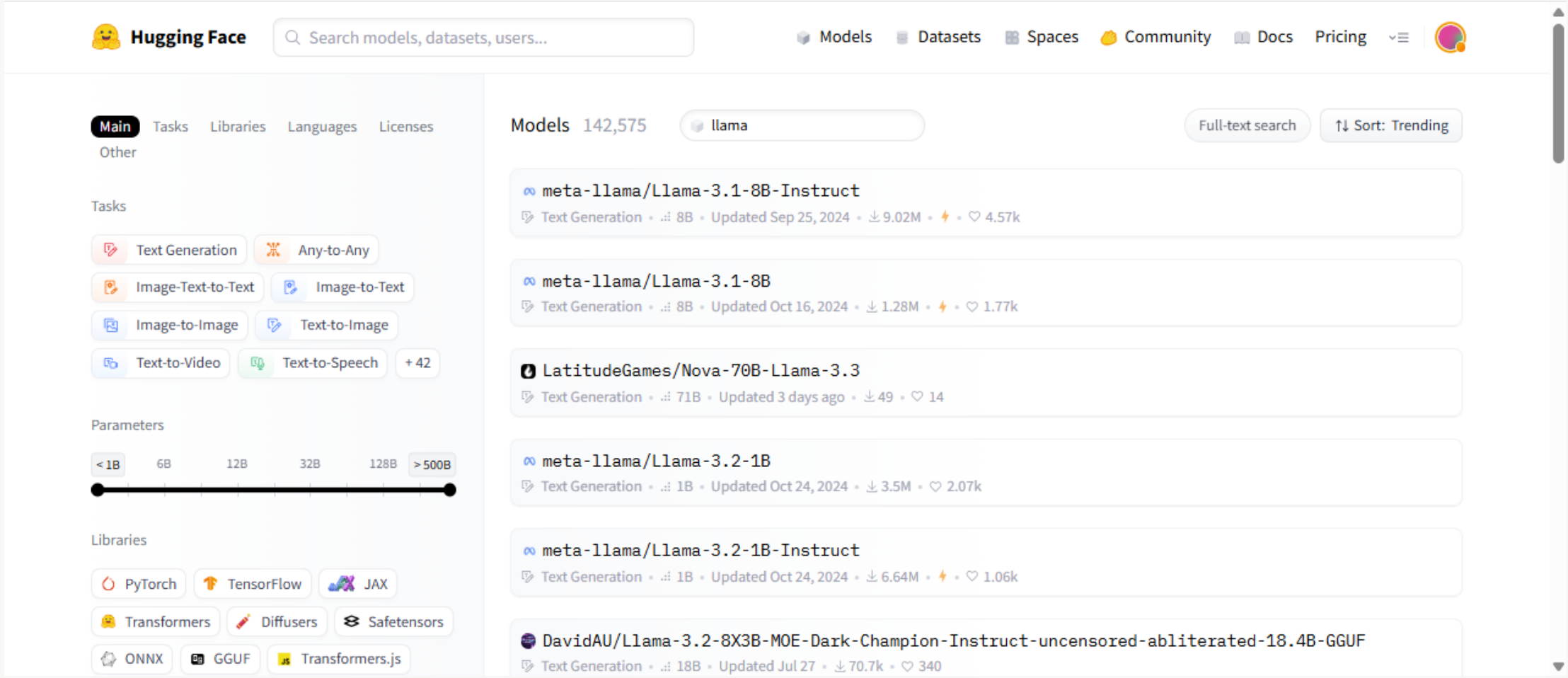
Gemma

Mistral

LLaMA

Open-weight LLMs

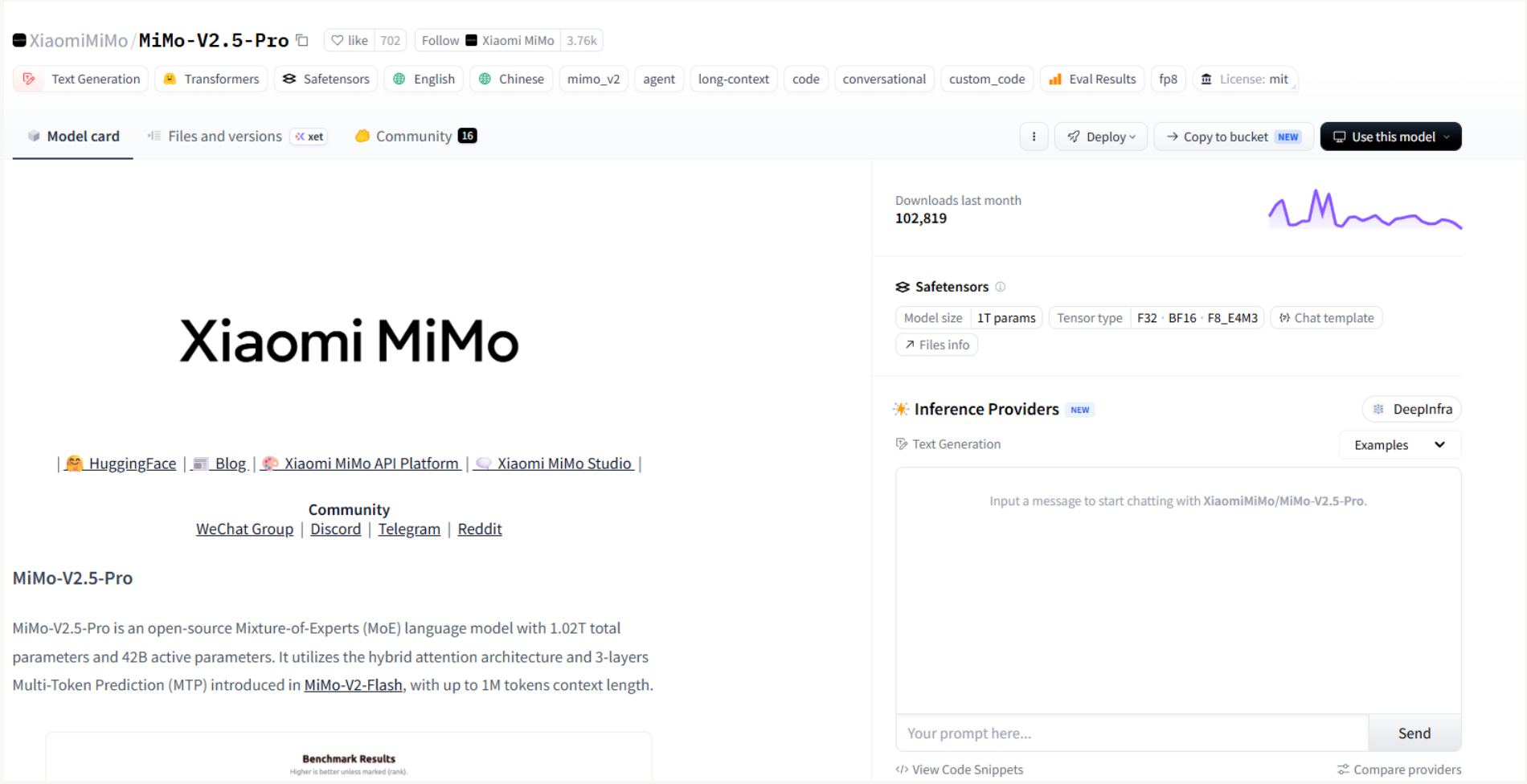
Hugging Face



https://huggingface.co/models

MiMo

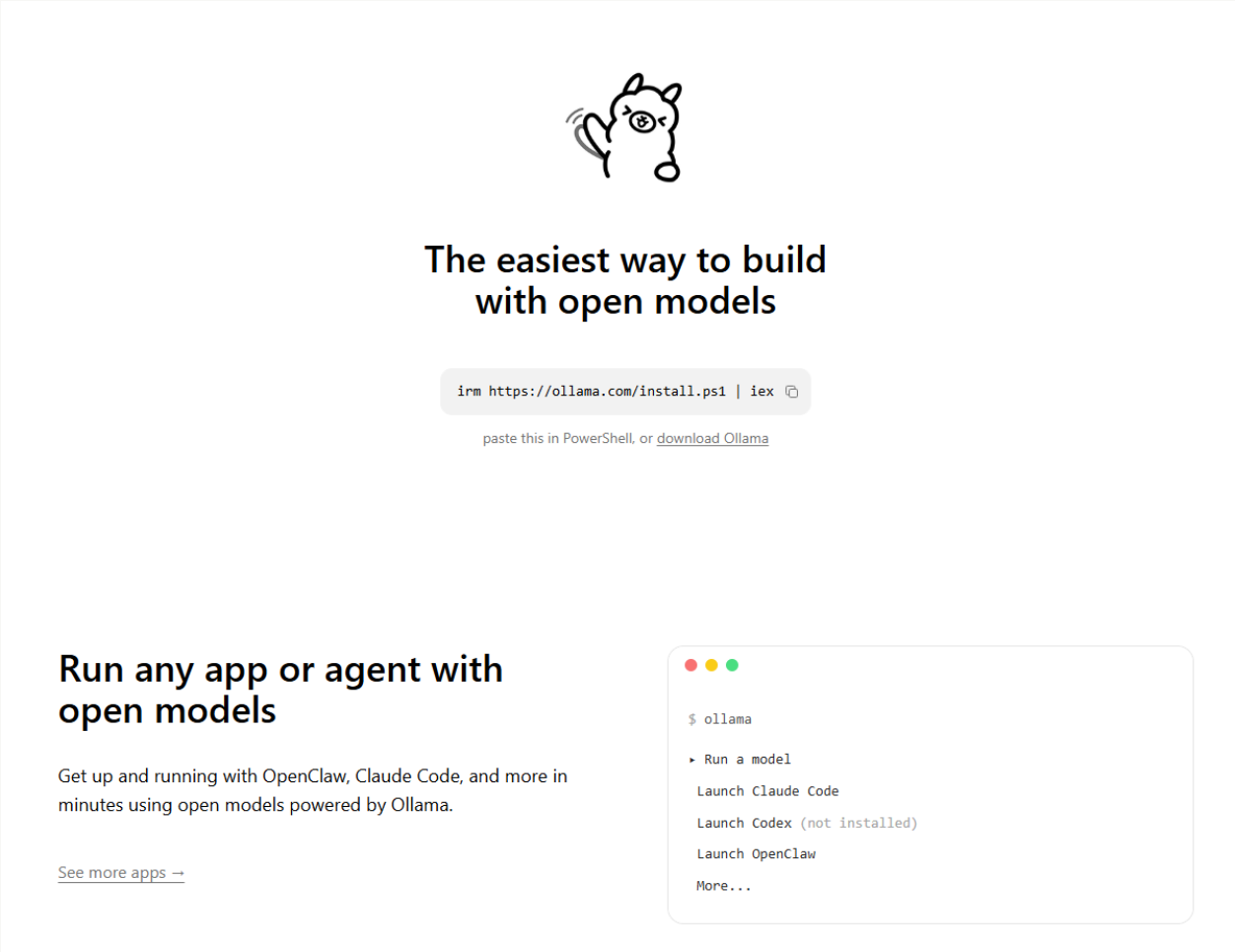
We will provide MiMo API for all students!



https://huggingface.co/XiaomiMiMo/MiMo-V2.5-Pro

Local Deployment

We will provide GLM API for all students!



https://ollama.com/

Recommendations

An excellent course of LLMs by Prof. Hung-yi Lee(李宏毅 台湾大学).

https://www.youtube.com/playlist?list=PLJV_el3uVTsMMGi5kbnKP5DrDHZpTX0jT

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Next lecture: Principles of Agents