

LLM — Use it and make it

Some slides borrow from CMU Roshan Sharma

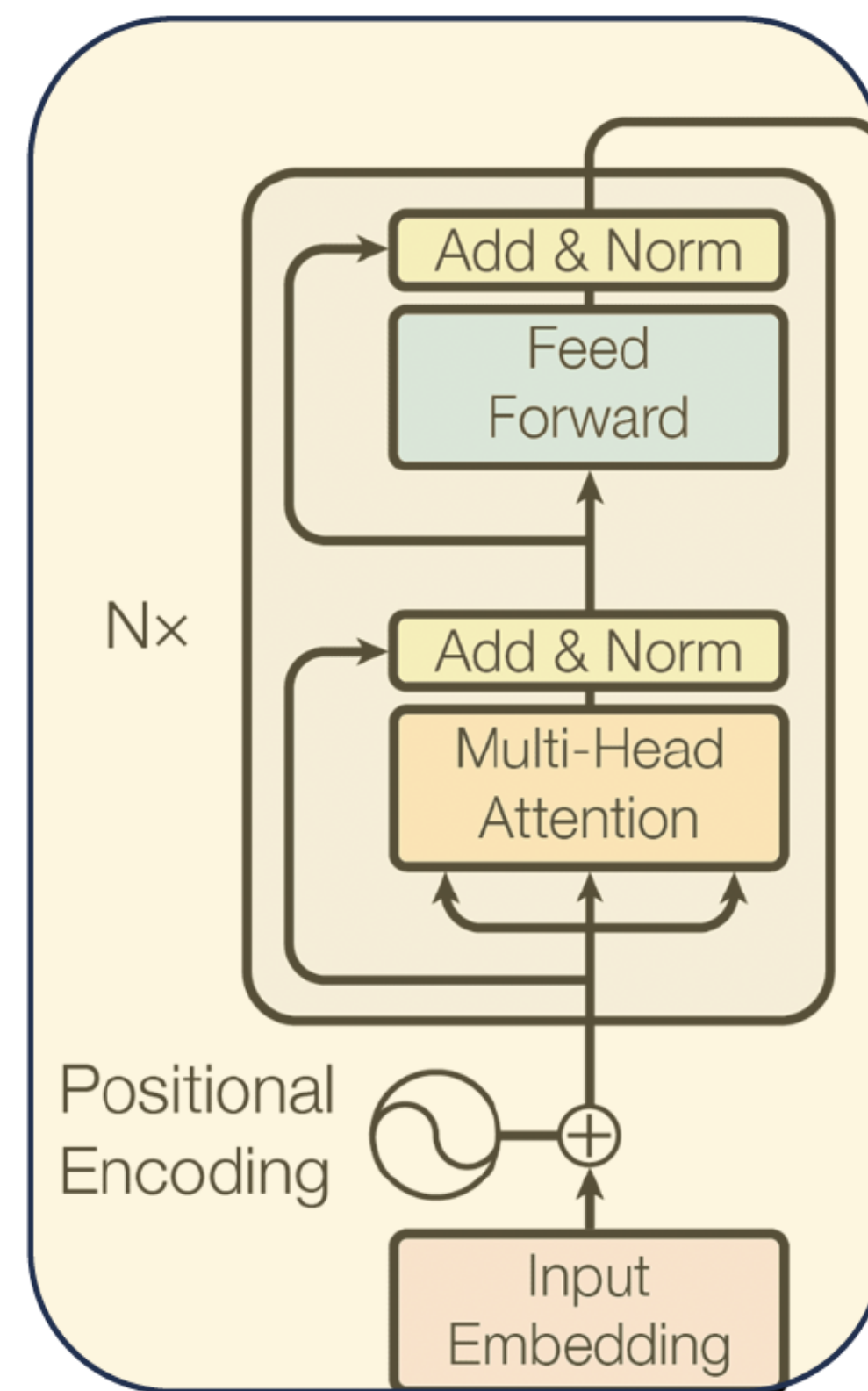
你是怎么使用GPT的？

- 你使用过哪些大语言模型？
- 你一般在哪些场景使用？
- 你有什么使用技巧？
- 你发现大语言模型有哪些优点和缺点？
- 你有关注过最新的关于大语言模型的研究吗？

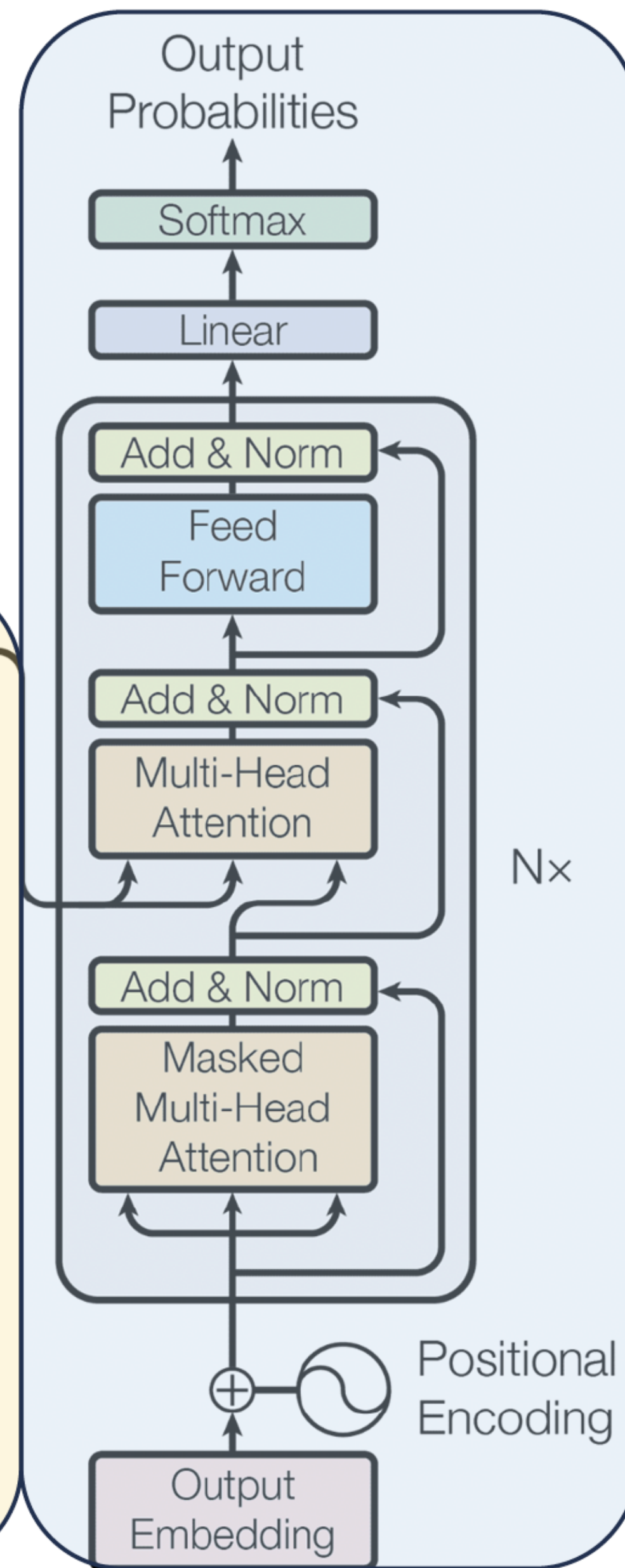
Background

BERT
Oct 2018

Representation



Inputs



Outputs
(shifted right)

GPT
Jun 2018

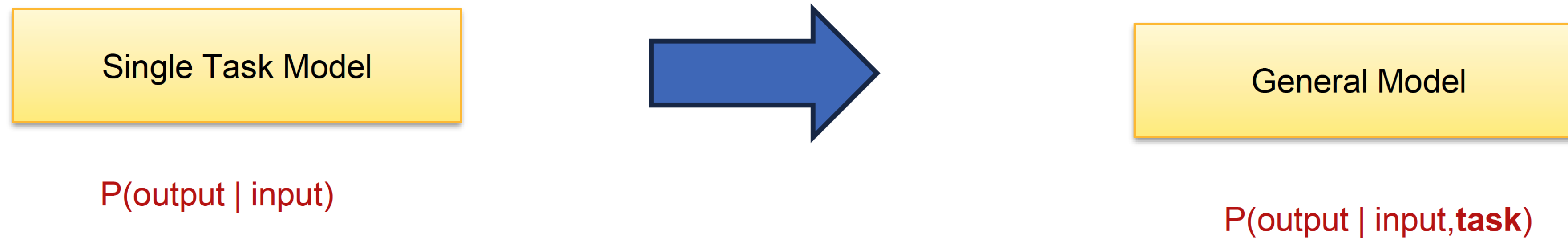
Generation

GPT VS BERT

特点	BERT (Encoder)	GPT (Decoder)
目标任务	表示学习（自然语言理解任务，如分类、问答）	生成任务（自然语言生成任务，如文本续写、对话）
上下文方向	双向：同时关注输入的前后上下文	单向：只能关注输入中之前的信息
注意力机制	多头注意力（Multi-Head Attention）	掩码多头注意力（Masked Multi-Head Attention）
输入与输出	输入 → 表示（特征向量）	输入 → 输出概率（预测下一个单词）
适用场景	情感分析、分类、命名实体识别、问答等理解任务	文本生成、续写、对话、代码生成等生成任务

GPT2 — for what?

- Primary shift comes from modeling assumptions from single-task to general model



- Task descriptions may be provided as text – for example, translate this French text to English

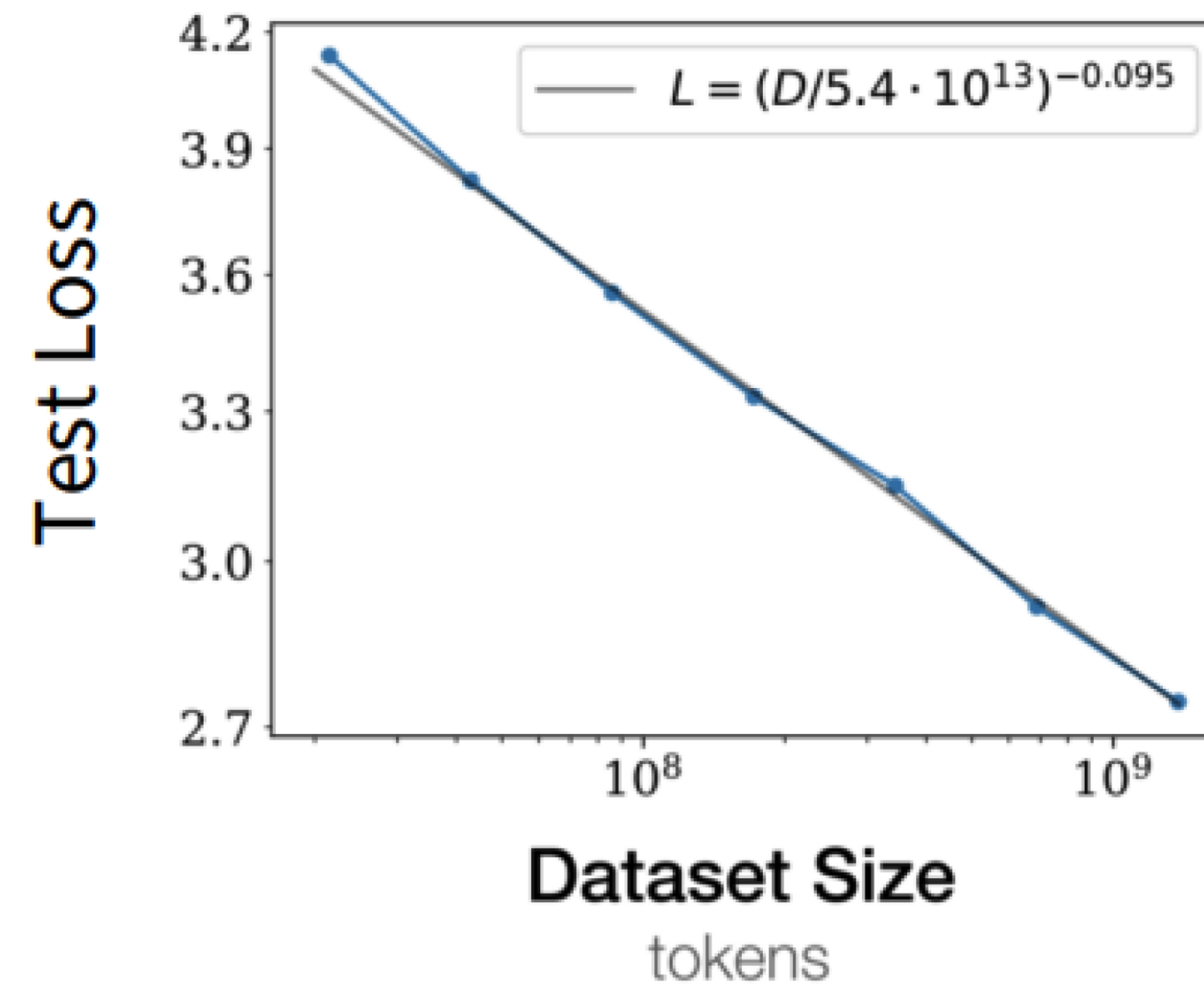
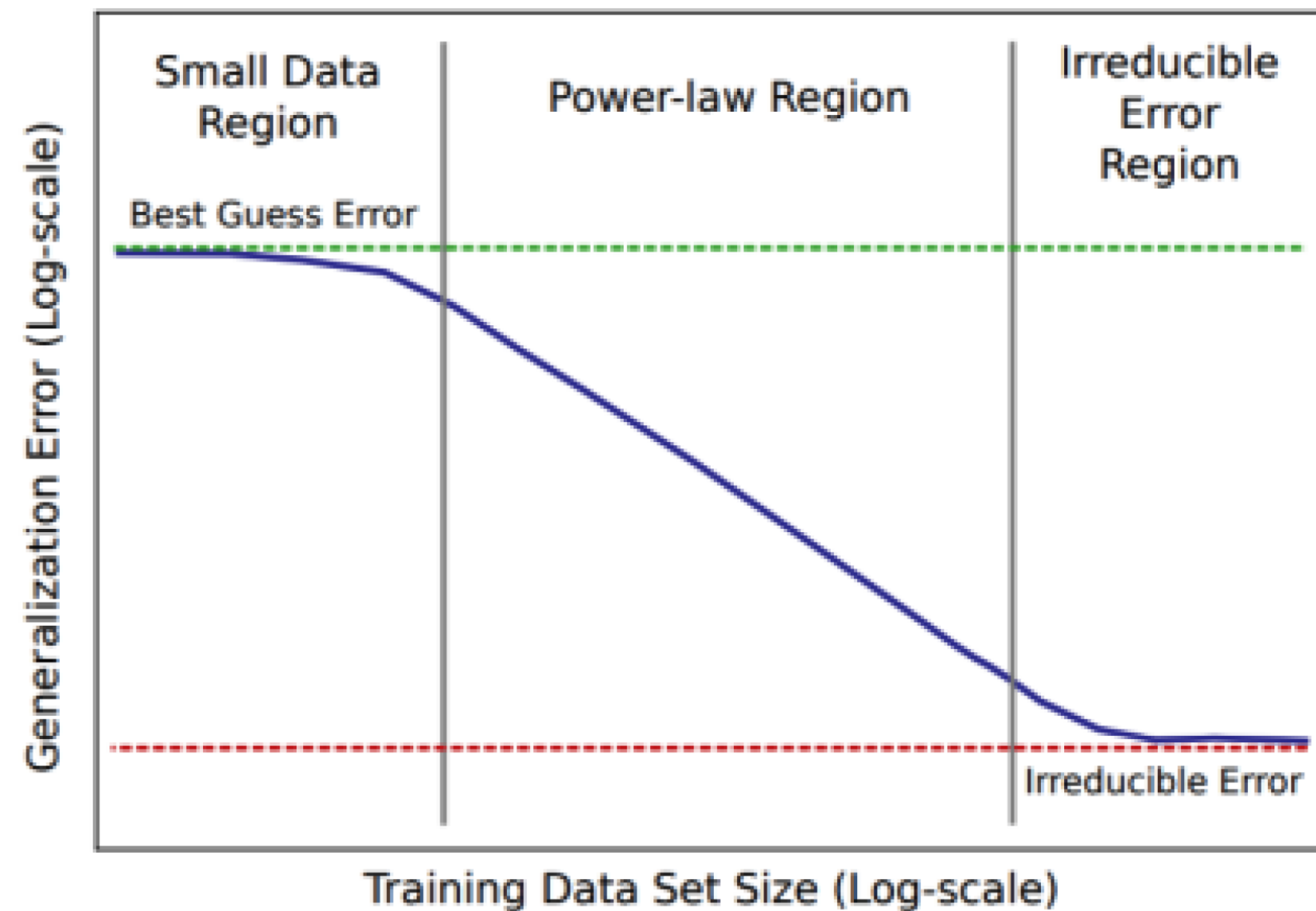
GPT2—How to work?

- Diverse training data
 - Model can do many disparate tasks with no training at all!
- Scaling model capacity and data

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	21.8
117M	35.13	45.99	87.65	83.4	29.41	65.85	1.16	1.17	37.50	75.20
345M	15.60	55.48	92.35	87.1	22.76	47.33	1.01	1.06	26.37	55.72
762M	10.87	60.12	93.45	88.0	19.93	40.31	0.97	1.02	22.05	44.575
1542M	8.63	63.24	93.30	89.05	18.34	35.76	0.93	0.98	17.48	42.16

GPT2—Data Scaling

- Loss and dataset size is linear on a log-log plot
- This is “power-law scaling”



GPT2 – Scaling Laws

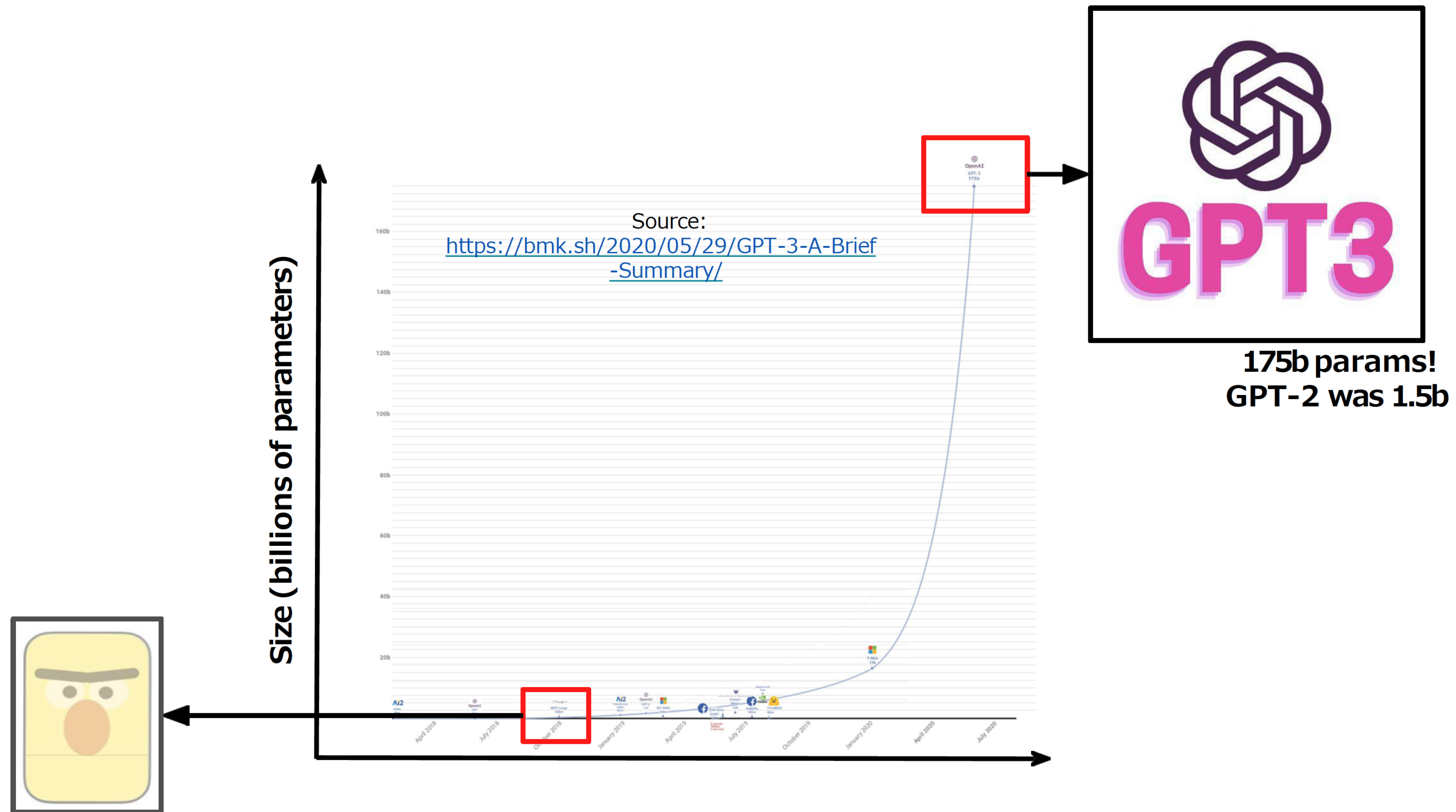
- Open AI Study : **Scaling Laws for Neural Language Models** ([Kaplan et al. 2020](#))
- Key Findings:
 - Performance depends strongly on scale, and weakly on the model shape
 - Larger models are more sample-efficient
 - Smooth power laws ($y = ax^k$) b/w empirical performance & N - parameters, D - dataset size, C - compute

Scaling Effects

- The effect of some hyperparameters on big LMs can be predicted before training – optimizer (Adam v/s SGD), model depth, LSTM v/s Transformer
- Idea:
 - Train a few smaller models
 - Establish a scaling law (e.g. ADAM vs SGD scaling law)
 - Select optimal hyper param based on the scaling law prediction

Milestone

Model Scaling



Emergent Abilities with GPT-3 – Wei et. al 2022

- Emergent abilities:
 - not present in smaller models but is present in larger models
 - Do LLMs like GPT3 have these ?
- Findings:
 - GPT-3 trained on text can do arithmetic problems like addition and subtraction
 - Different abilities “emerge” at different scales
 - **Model scale is not the only contributor to emergence** – for 14 BIG-Bench tasks, LaMDA 137B and GPT-3 175B models perform at near-random, but PaLM 62B achieves above-random performance
 - Problems LLMs can’t solve today may be emergent for future LLMs

Training of Decoder-only LLMs – Llama 2

1. Auto-regressive Pre-training - Train to predict the next token on very large scale corpora (~3 trillion tokens)
2. Instruction Fine-tuning/ Supervised Fine-tuning (SFT) - Fine-tune the pre-trained model with pairs of (instruction+input,output) with large dataset and then with small high-quality dataset

Instruction fine-tuning provides as a prefix a natural language description of the task along with the input.

- E.g. Translate into French this sentence: my name is -> je m'appelle

Llama 2 Architecture (Ouyang et. al.)

- Decoder-only model
- Changes in transformer module:
 - Norm after sublayer -> Norm before sublayer
 - LayerNorm -> RMSNorm for stability
 - Activation: ReLU -> SwiGLU(x) = Swish(xW) xV = xW Sigmoid(AxW) xV
 - Position Embedding: Absolute/Relative -> RoPE (Rotary PE)
 - Long contexts : Multi-head attention -> Grouped-query attention

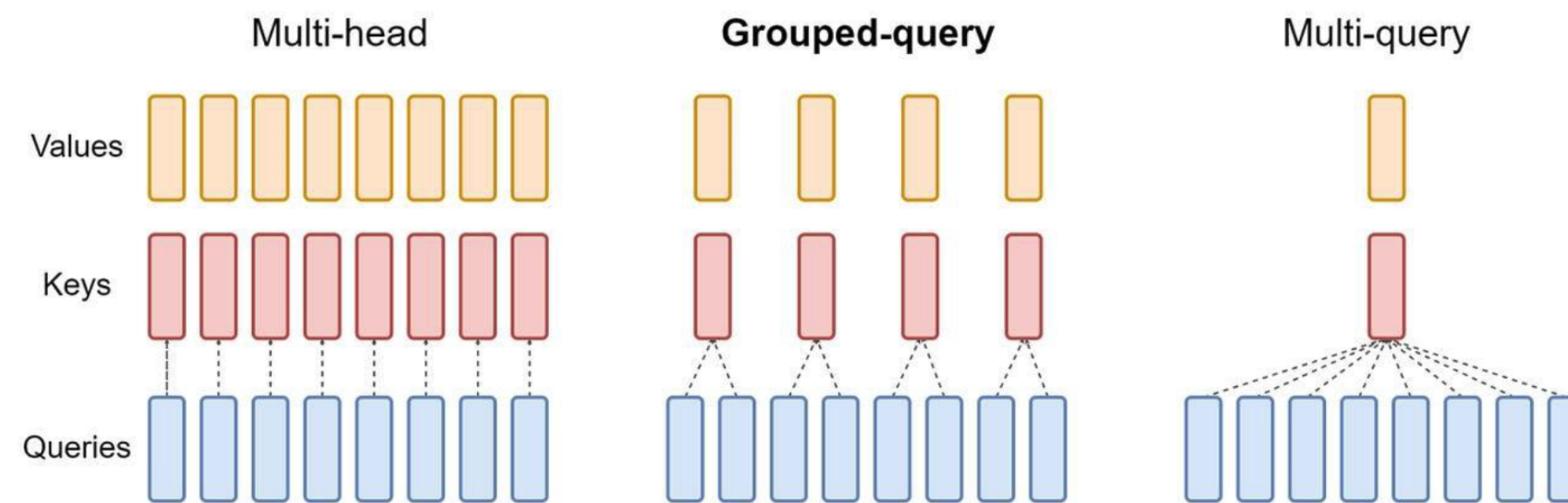


Figure 2: Overview of grouped-query method. Multi-head attention has H query, key, and value heads. Multi-query attention shares single key and value heads across all query heads. Grouped-query attention instead shares single key and value heads for each *group* of query heads, interpolating between multi-head and multi-query attention.

Instruction Tuning ([Wei et. al. 2021](https://openreview.net/pdf?id=gEZrGCozdqR))

Finetune on many tasks (“instruction-tuning”)

Input (Commonsense Reasoning)

Here is a goal: Get a cool sleep on summer days.
How would you accomplish this goal?
OPTIONS:
-Keep stack of pillow cases in fridge.
-Keep stack of pillow cases in oven.

Target
keep stack of pillow cases in fridge

Input (Translation)

Translate this sentence to Spanish:
The new office building was built in less than three months.

Target
El nuevo edificio de oficinas se construyó en tres meses.

Sentiment analysis tasks

Coreference resolution tasks

...



Inference on unseen task type

Input (Natural Language Inference)

Premise: At my age you will probably have learnt one lesson.
Hypothesis: It's not certain how many lessons you'll learn by your thirties.
Does the premise entail the hypothesis?
OPTIONS:
-yes -it is not possible to tell -no

FLAN Response
It is not possible to tell

LLM Inference: Prompting

- Prompts
 - Tell the model what to do in natural language
 - For example, generate a textual summary of this paragraph:
 - Can be as short or long as required
- Prompt Engineering
 - The task of identifying the correct prompt needed to perform a task
 - General rule of thumb be as specific and descriptive as possible
 - Can be manual or automatic (prefix-tuning, paraphrasing etc.)

In-context learning/ Few-shot prompting (Brown,21)

- Provide a few examples along with the instruction

Instruction | Please classify movie reviews as 'positive' or 'negative'.

Examples

| Input: I really don't like this movie.
| Output: negative

| Input: This movie is great!
| Output: positive

Chain of thought prompting (Wei, 2021)

- Get the model to work through the steps of the problem

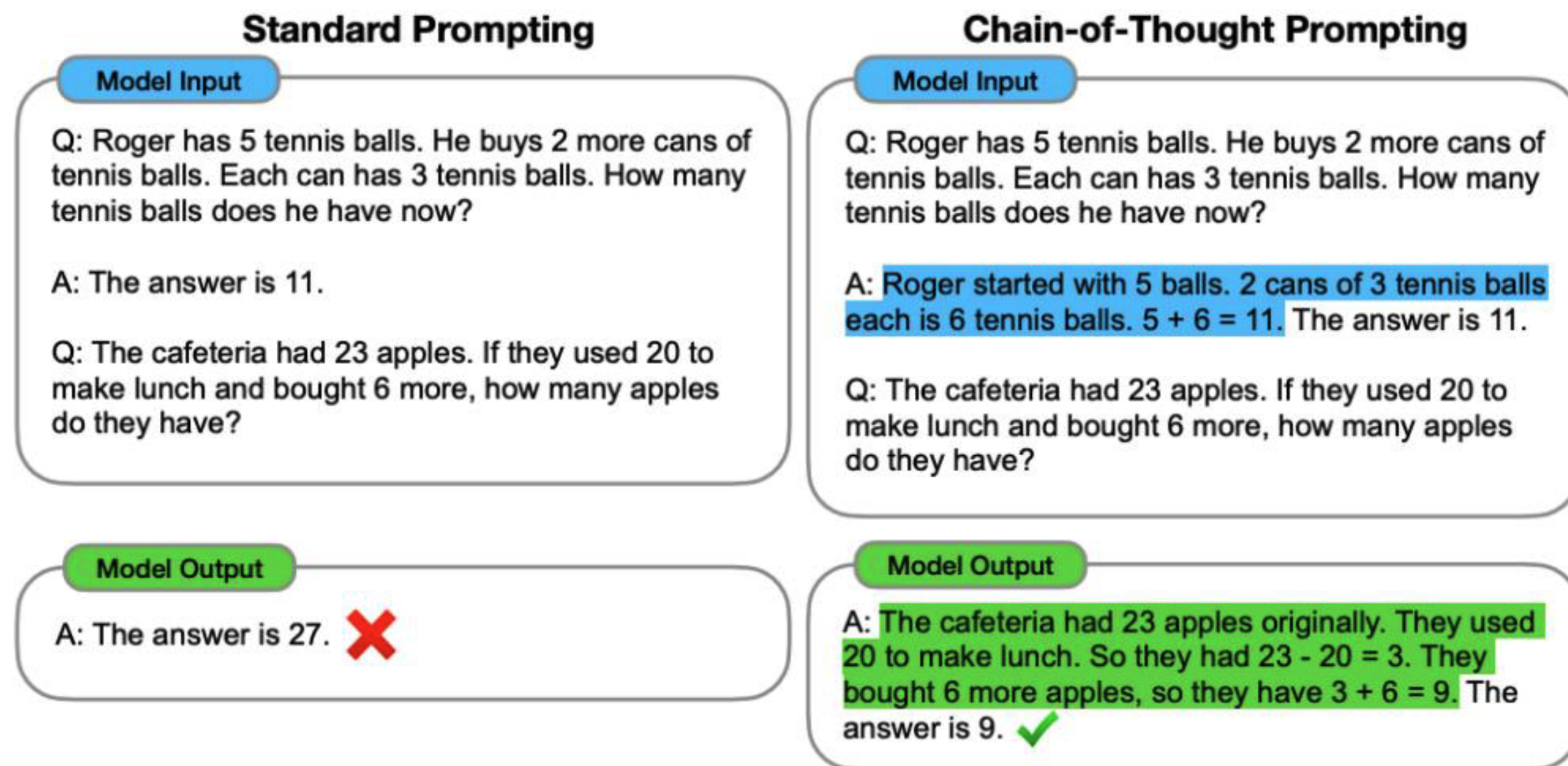


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

Core Tech

- Scaling (Emergent Abilities)
- Training (Parallelism)
- Ability Eliciting (CoT)
- Alignment tuning (RLHF)
- Tool manipulation (More than language)

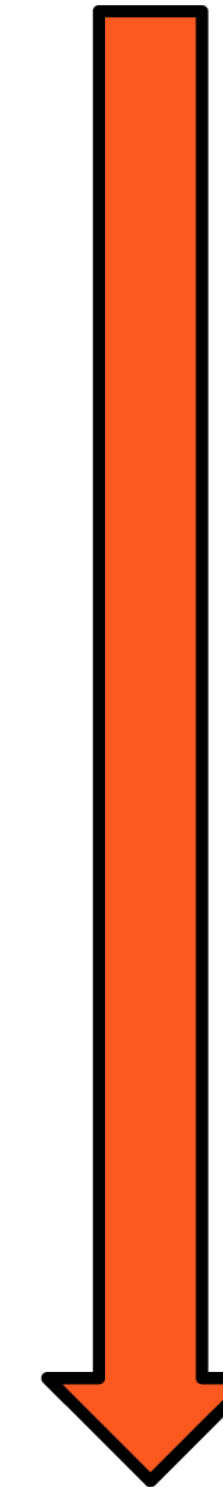
Core Ability

- In-context learning
 - Zero-shot
 - Few-shot
- Instruction following
 - Instruction tuning
- Step-by-step reasoning
 - Chain of thoughts

What to Pick?

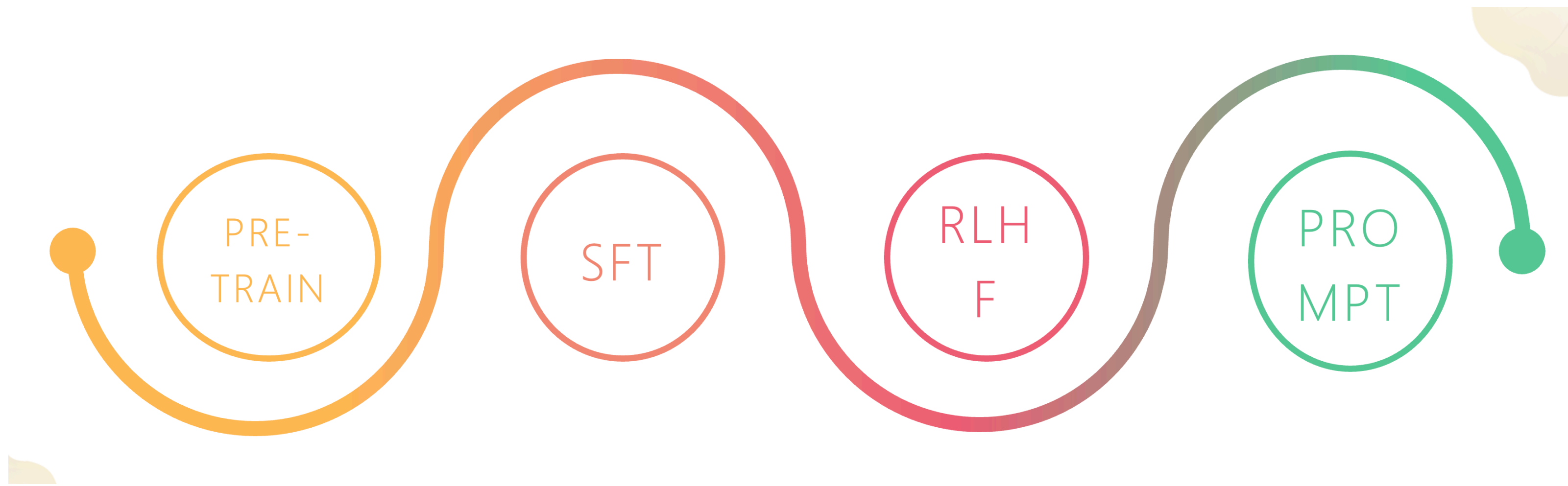
1. Full Fine-tuning (FT)
 - a. +Strongest performance
 - b. - Need curated and labeled dataset for each new task (typically 1k-100k+ ex.)
 - c. - Poor generalization, spurious feature exploitation
2. Few-shot (FS)
 - a. +Much less task-specific data needed
 - b. +No spurious feature exploitation
 - c. - Challenging
3. One-shot (1S)
 - a. +"Most natural," e.g. giving humans instructions
 - b. - Challenging
4. Zero-shot (0S)
 - a. +Most convenient
 - b. - Challenging, can be ambiguous

Stronger
task-specific
performance



More convenient,
general, less data

Flow



- Pre-train: 语料库, 多参数
- SFT: 特性化, 专门化
- RLHF: 对齐人类认知
- Prompt: 充分利用模型

Open Challenges - LLMs

- **New Capabilities**
 - Multimodal
 - Multi-lingual
 - More Complex Tasks
- **Performance**
 - Reduce Hallucinations
 - Improve Alignment with Human Preference
 - Increase Context Length Efficiently
 - Improve Data, Training Strategy, and Model Architecture
- **Efficiency**
 - Computational cost, time, and money
 - Compute architecture – GPU/ TPU/ HPU

Open Challenges - LLMs

- Safety
 - Reduce Harm
 - Improve Adversarial Robustness
 - Privacy Concerns
- Interpretability
 - Why do LLMs do what they do?

Summary

- LLMs are large-scale models that possess astounding abilities
- Scaling both data and model capacity is important for performance and leads to the emergence of new abilities
- Decoder-only architectures are popular for convergence and performance
- LLMs are trained using pre-training, SFT, RLHF
- LLMs are evaluated using prompting/ strategies like ICL and CoT
- Multimodal LLMs can process audio, text, images and more.

Example

Requirements

- 如何利用大语言模型生成编译器的测试用例：
 - 考虑编译器本身可以利用的有哪些？
 - 考虑之前提到的使用LLM的方法
 - 考虑如何用尽量少的token（gpt api是根据token计费）实现

OOPSLA'24 WhiteFox

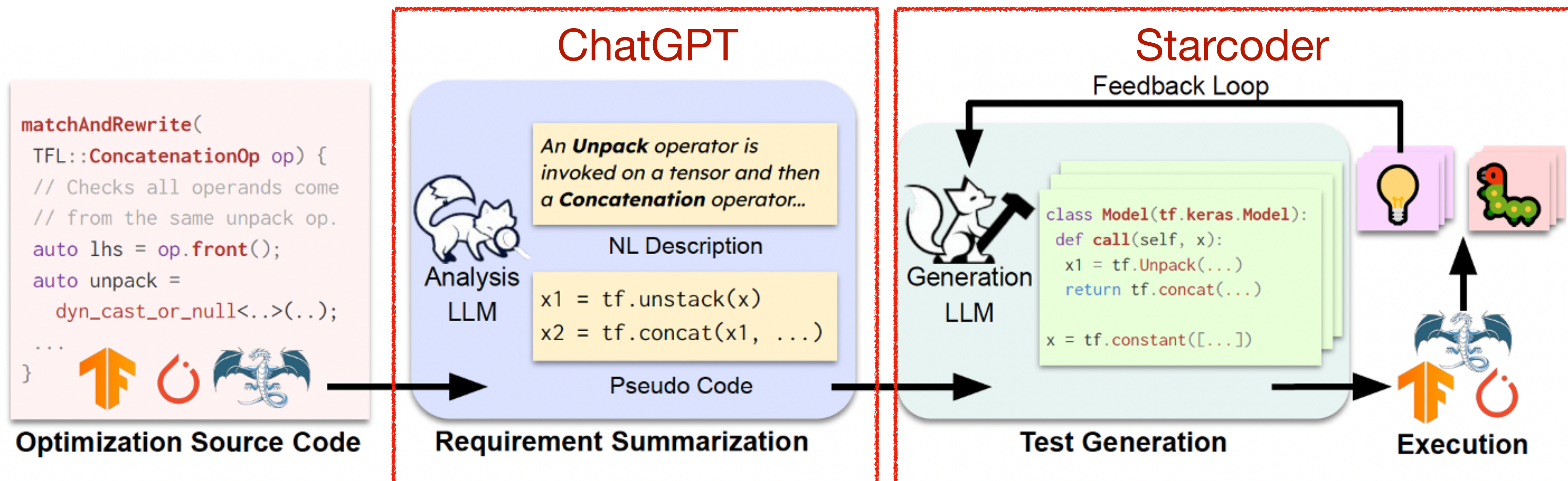
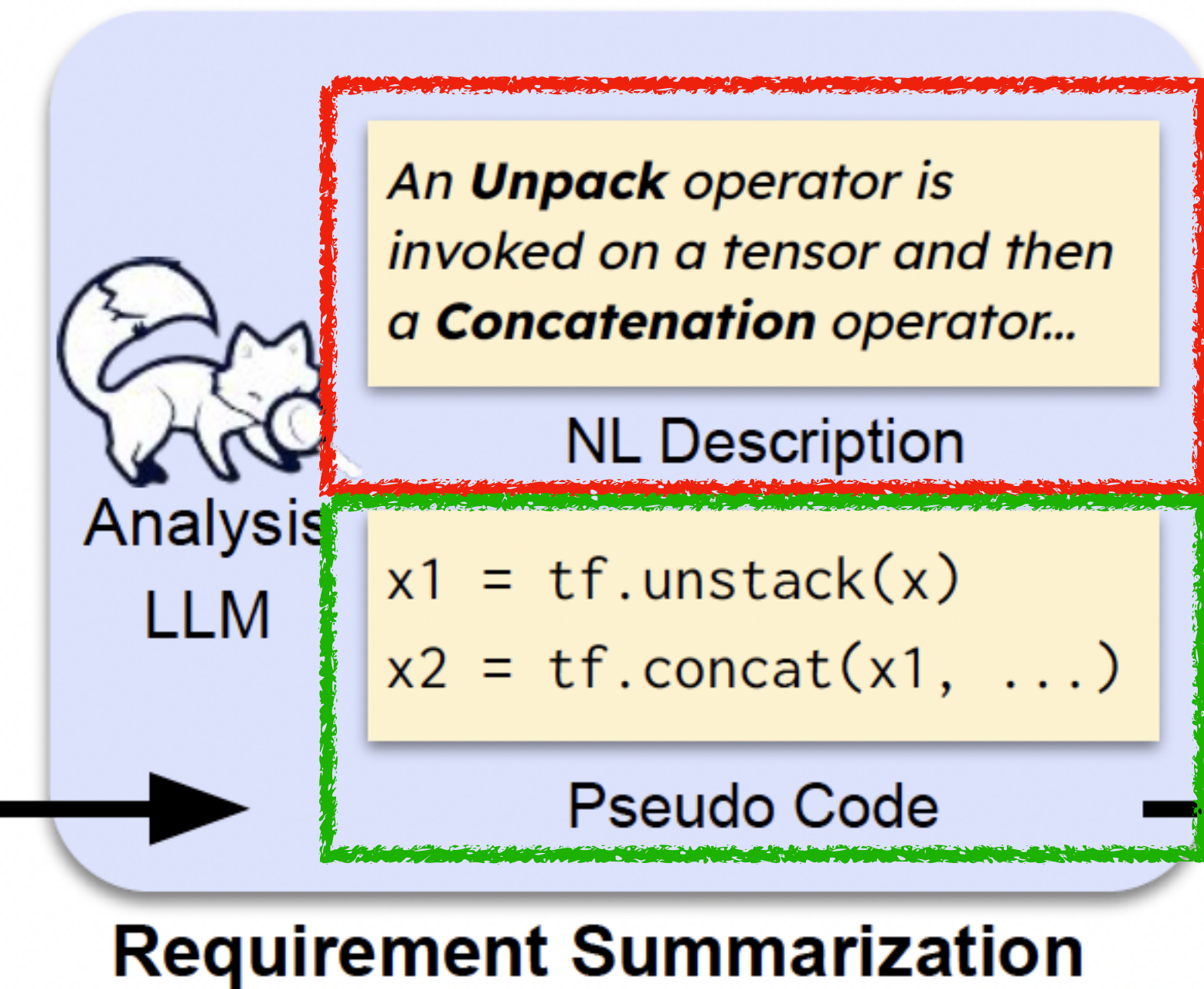
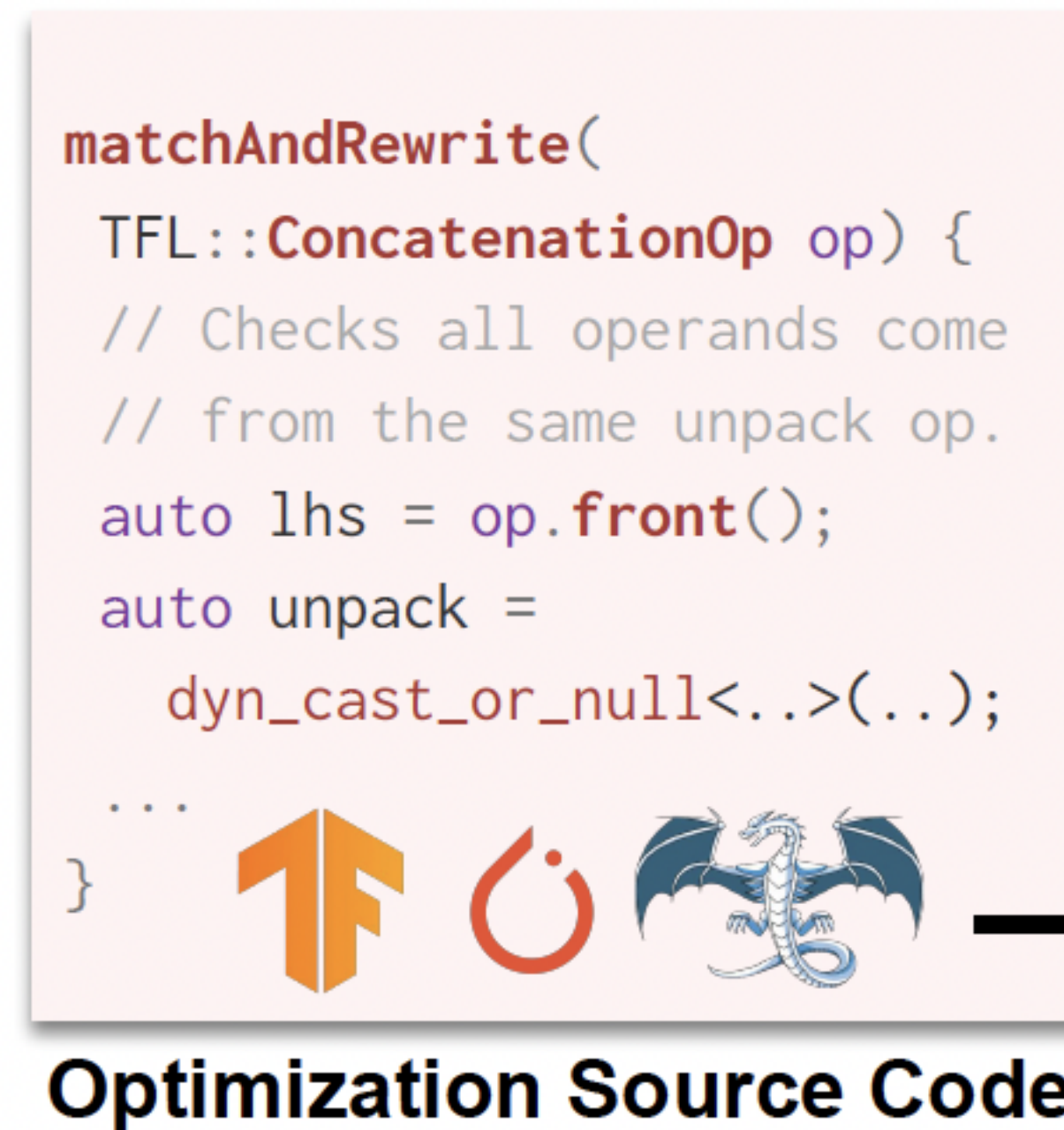


Fig. 2. Overview of WhiteFox.

双语言模型框架：需求总结和测试生成

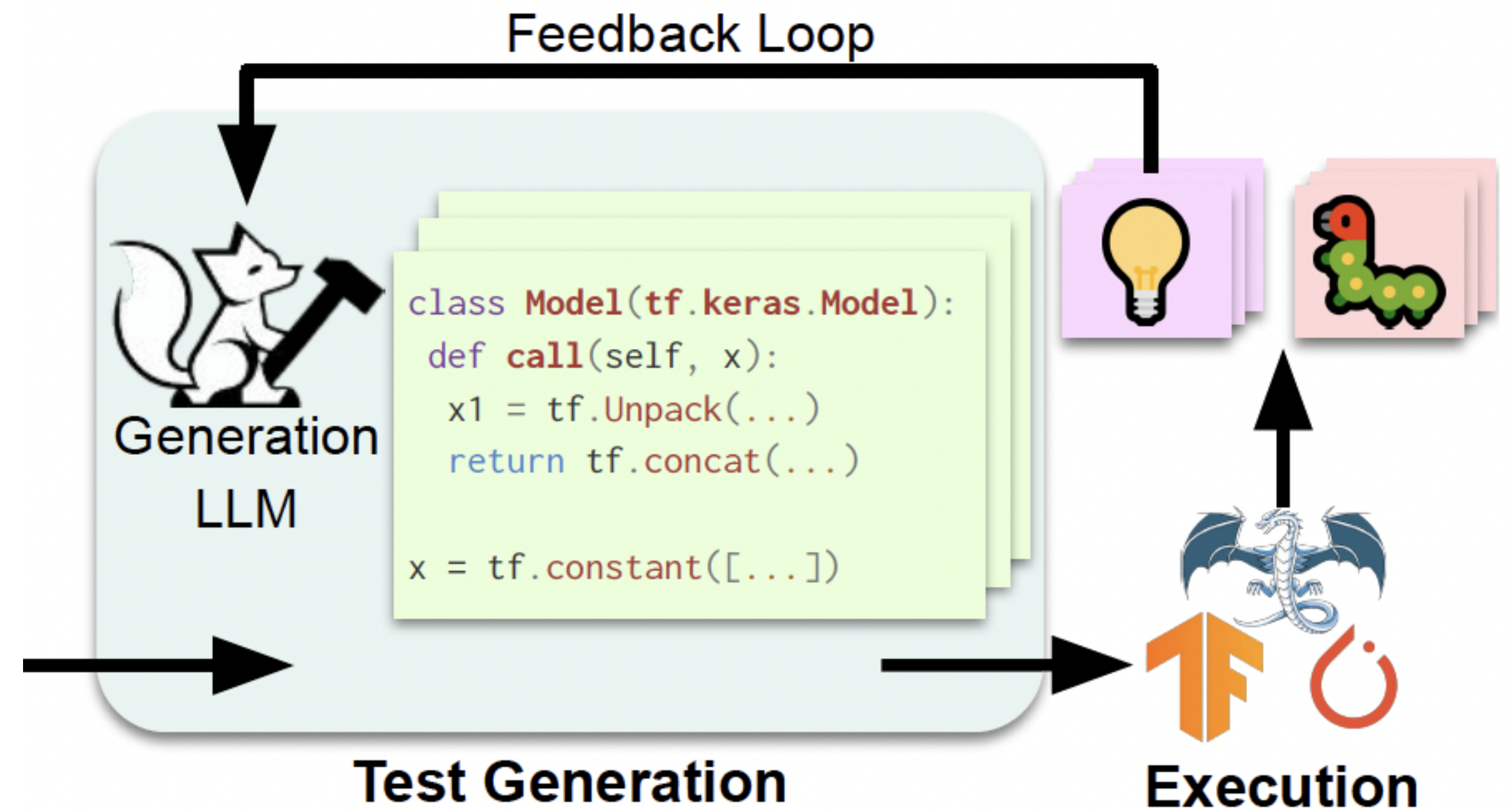
Requirement Summarization

- Input: Optimization Source Code & Helper Functions
- Output: Requirements of how the optimization is triggered
- How to describe the requirement?
 - NL & Pseudocode



测试生成

- Input: Requirements generated by GPT
- Output: Testing code
- 反馈机制
 - 成功样本驱动的提示优化
 - 提高触发优化的权重



Prompt for GPT

1. 例子 (问)

```
1 ### Please describe the C++ code that can trigger the optimization pass
`DeadArgumentEliminationPass` in LLVM. The C++ code can used to reach out `F.replaceAllUsesWith
(ConstantExpr::getBitCast(NF, F.getType()));` line and make the boolean variable `Changed` equal
to `True`. The description should be concise and clear. Use code to illustrate patterns or
constraints as needed. Please only describe the characteristics of the C++ code. Do not describe
the function code or what happens after the optimization is triggered. Do not include the
optimization name and target line information in your description. Your description must have the
`main` function.
```

```
# The code of the optimization and its helper functions:

```cpp
PreservedAnalyses DeadArgumentEliminationPass::run(Module &M,
 ModuleAnalysisManager &) {
 bool Changed = false;

 // First pass: Do a simple check to see if any functions can have their "...
 // removed. We can do this if they never call va_start. This loop cannot be
 // fused with the next loop, because deleting a function invalidates
 // information computed while surveying other functions.
 LLVM_DEBUG(dbgs() << "DeadArgumentEliminationPass - Deleting dead varargs\n");
 for (Function &F : llvm::make_early_inc_range(M))
 if (F.getFunctionType()->isVarArg())
 Changed |= deleteDeadVarargs(F);

 if (!Changed)
 return PreservedAnalyses::all();
 return PreservedAnalyses::none();
}
```

1. NL—Req需要满足的条件
2. Pseudocode — 优化源代码以及相关代码

Question:

1. Same example for all the prompt
2. The reason for choosing this example
3. The best form of Req



# Prompt for GPT

## 1. 例子 (答)

```
68 # Description
69 The C++ code should contain the following pattern:
70 ```
71 // no va_arg in stdarg.h is used
72
73 // define one function which has variable arguments
74 static int varargs_func(int x, ...) {
75 RANDOM_CODE; // Do something here with input arguments, but don't generate the codes contain
76 va_args
77 int result = x OPERATOR ANY_VALUE; // result is related to x
78 return result;
79 }
80
81 int main() {
82 RANDOM_CODE; // declare some variables
83 int x = ANY_VALUE; // declare variable x
84 ANY_TYPE y = ANY_VALUE; // declare y
85 ANY_TYPE z = ANY_VALUE; // declare z
86 // invoke the func varargs_func
87 int result = varargs_func(x, y, z, ANY_VALUE, ANY_VALUE); // you can pass the arguments with
88 any number, any type
89 return result;
90 }
91 ```
```

# Prompt for GPT

## 2. 要求

```
93 ### Please describe the C++ code that can trigger the optimization pass `{Passname}` in LLVM. The
 C++ code can used to reach out `{target_line}` line and make the boolean variable `Changed` equal
 to `True`. The description should be concise and clear. Use code to illustrate patterns or
 constraints as needed. Please only describe the characteristics of the C++ code. Do not describe
 the function code or what happens after the optimization is triggered. Do not include the
 optimization name and target line information in your description. Your description must have the
 `main` function.
94
95
96 # The code of the optimization and its helper functions:
97
98 ```
99 {source_llvm}
100 ```
101
102 # Description
```

1. Replace by different pass code

2. Description to be continued by GPT

# Prompt for GPT

## 3. 生成的Req

The C program should contain the following pattern:

```
```c
#include <stdio.h>

// declare variables
int a = 10; // declare and initialize a
int b = 20; // declare and initialize b
int c = 0;  // declare and initialize c

// define a function that performs some operations
int compute(int x, int y) {
    int result = x + y; // perform an operation
    return result; // return the result
}

int main() {
    int main_result = 0; // declare the result

    // perform some operations
    c = a + b; // perform an operation that is not used later

    // invoke the compute function
    main_result = compute(a, b);

    // print the result
    printf("Result: %d\n", main_result);

    return 0; // return the result
}
```
```

This pattern characterizes scenarios where within the `main` function, there exists a computation or operation

# Prompt for StarCoder

## 1. 例子 (问)

```
Please generate a valid C++ Code that meets the requirements below. The code should contain a `main` function. And the main function gives back an output value. Please initialize all the variables you define with a value. Please do not include any undefined behavior in your code.
```

```
Description of requirements

The C++ code should contain the following pattern:
...
// no va_arg in stdarg.h is used

// define one function which has variable arguments
static int varargs_func(int x, ...) {
 RANDOM_CODE; // Do something here with input arguments, but don't generate the codes
 contain va_args
 int result = x OPERATOR ANY_VALUE; // result is related to x
 return result;
}

int main() {
 RANDOM_CODE; // declare some variables
 int x = ANY_VALUE; // declare variable x
 ANY_TYPE y = ANY_VALUE; // declare y
 ANY_TYPE z = ANY_VALUE; // declare z
 // invoke the func varargs_func
 int result = varargs_func(x, y, z, ANY_VALUE, ANY_VALUE); // you can pass the arguments
 with any number, any type
 return result;
}
...
```

1. **NL**—生成的代码需要满足的条件
2. **Pseudocode** —Req

Question:

1. Same example for all the prompt
2. The reason for choosing this example

# Prompt for StarCoder

## 1. 例子 (答)

```
C++ Code begins

```cpp
#include <vector>
#include <string>

static int varargs_func(int x, ...) {
    int result = 10;
    if (x % 2) {
        result = x * 32 + 15;
    } else {
        result = x - 15;
    }
    if (x > 100) {
        result = result * 2;
    } else {
        result = result * 2;
    }
    return result;
}

int main() {
    int result = 0;
    int a = (1 << 31);
    std::vector<int> vec = {0, 1, 2, 3};
    std::string str = "hello";
    vec.assign(5, 42);
    int b = varargs_func(a, 18, 25, vec[1], 1.000000, str[3]);
    result = b;
    return result;
}
```
```

# Prompt for GPT

## 2. 要求

```
C++ Code ends
```

```
Please generate a valid C++ Code that meets the requirements below. The code should contain a `main` function. And the main function gives back an output value. Please initialize all the variables you define with a value.
```

```
Description of requirements
```

```
{Description}
```

```
C++ Code begins
```

Replace by Req generated by GPT

# Prompt for GPT

## 3. 生成的Test code

```
```cpp
#include <vector>
#include <string>

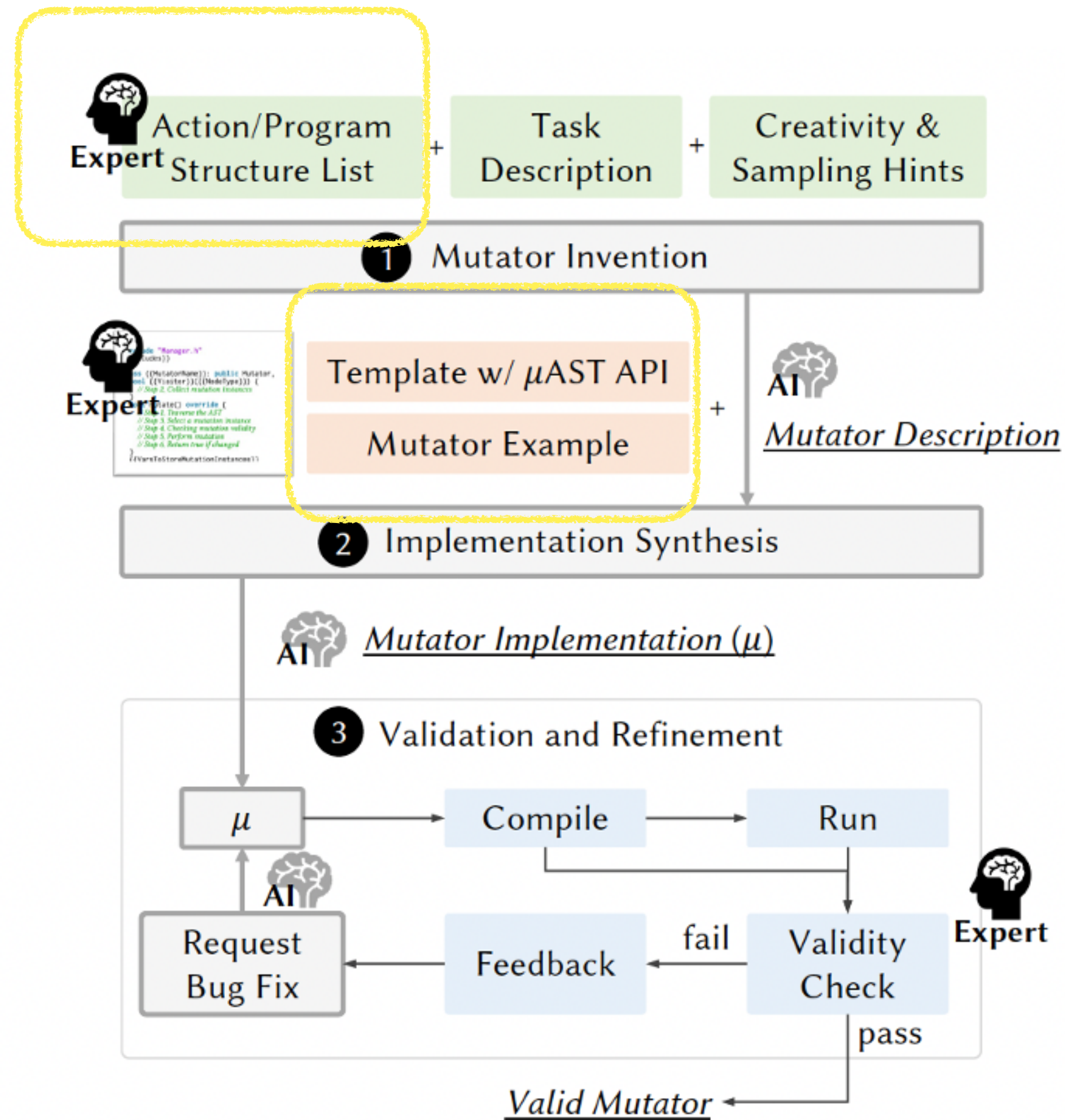
int compute(int x, int y) {
    int result = x + y;
    return result;
}

int main() {
    int main_result = 0;
    int a = (1 << 31);
    std::vector<int> vec = {0, 1, 2, 3};
    std::string str = "hello";
    int b = compute(a, 18);

    main_result = b;

    return main_result;
}
```
```

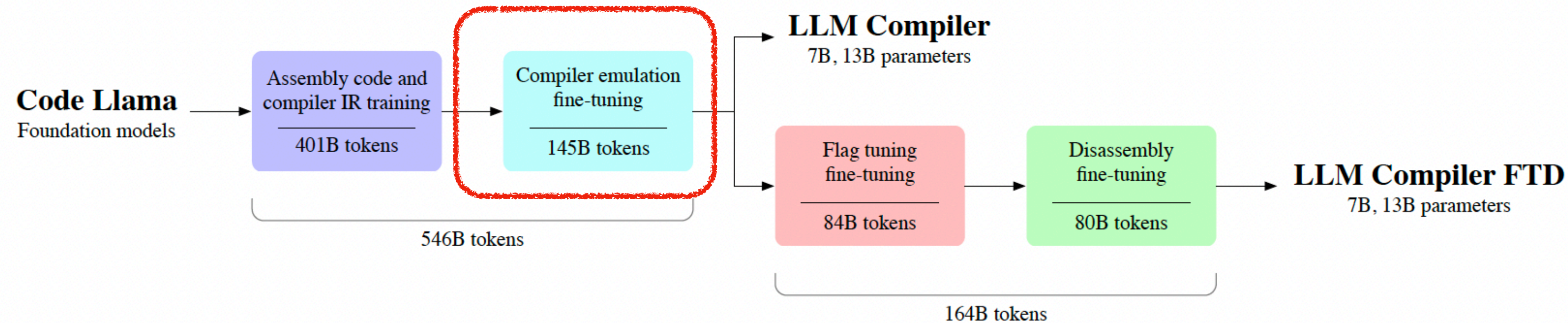
# METAMUT



- 提示中直接提供一个包含操作和程序结构的列表，操作包括Clang AST和IR API的成员函数。结构包含Clang AST节点类型。（让GPT可以选词填空）—生成Req
- 利用CoT指导实现，为LLM提供一个程序模板和分步指令。—实现
- 利用CoT进行生成测试用例、自我修正、验证。—验证反馈



# META LLM Compiler



- 利用通过大量汇编和 IR 数据增强对代码优化进行**训练**。
- **仿真微调**：学习【未经优化 to 优化后的 IR 或汇编代码】，从而有效地模拟编译器的优化过程
- 下游任务：**生成最优编译器标志**和**从汇编代码逆向生成 LLVM-IR**

**谢谢!**

**to be continued....**