AiDD AI+研发数字峰会
AI+ Development Digital summit
第5届 5

业务领域代码大模型深度探索与实践

顾小东 | 上海交通大学

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 · 敦煌站
K+ 思考周®研习社
时间：2025.08.29-30

K+峰会 · 上海站
K+ 金融专场
时间：2025.10.17-18

K+峰会 · 香港站
K+ 思考周®研习社
时间：2025.11.25-26

K+峰会详情

AiDD峰会

AiDD峰会 · 上海站
AI+研发数字峰会
时间：2025.05.17-18

AiDD峰会 · 北京站
AI+研发数字峰会
时间：2025.08.08-09

AiDD峰会 · 深圳站
AI+研发数字峰会
时间：2025.11.28-29

AiDD峰会详情

演讲嘉宾

顾小东

上海交通大学副教授、博士生导师

长期从事智能软件工程领域的研究工作，包括代码大模型、程序自动生成、代码翻译、代码搜索等。提出首个基于深度学习的代码搜索引擎DeepCS。在ICSE、FSE、ASE、TOSEM等顶级学术会议和期刊上发表学术论文30余篇，主持和参与多项国家自然科学基金、国家重点研发计划、国防课题等。并与华为、宁德时代、腾讯等企业开展广泛的产学研合作。

研究概览

代码生成

- 特定领域代码生成
- 仓库级代码生成
- 生成代码的修复

代码翻译

- 小样本代码翻译
- 代码翻译的细粒度评估
- C2Rust翻译

代码大模型

代码修复

- 大模型生成代码修复
- 仓库级问题修复

代码搜索

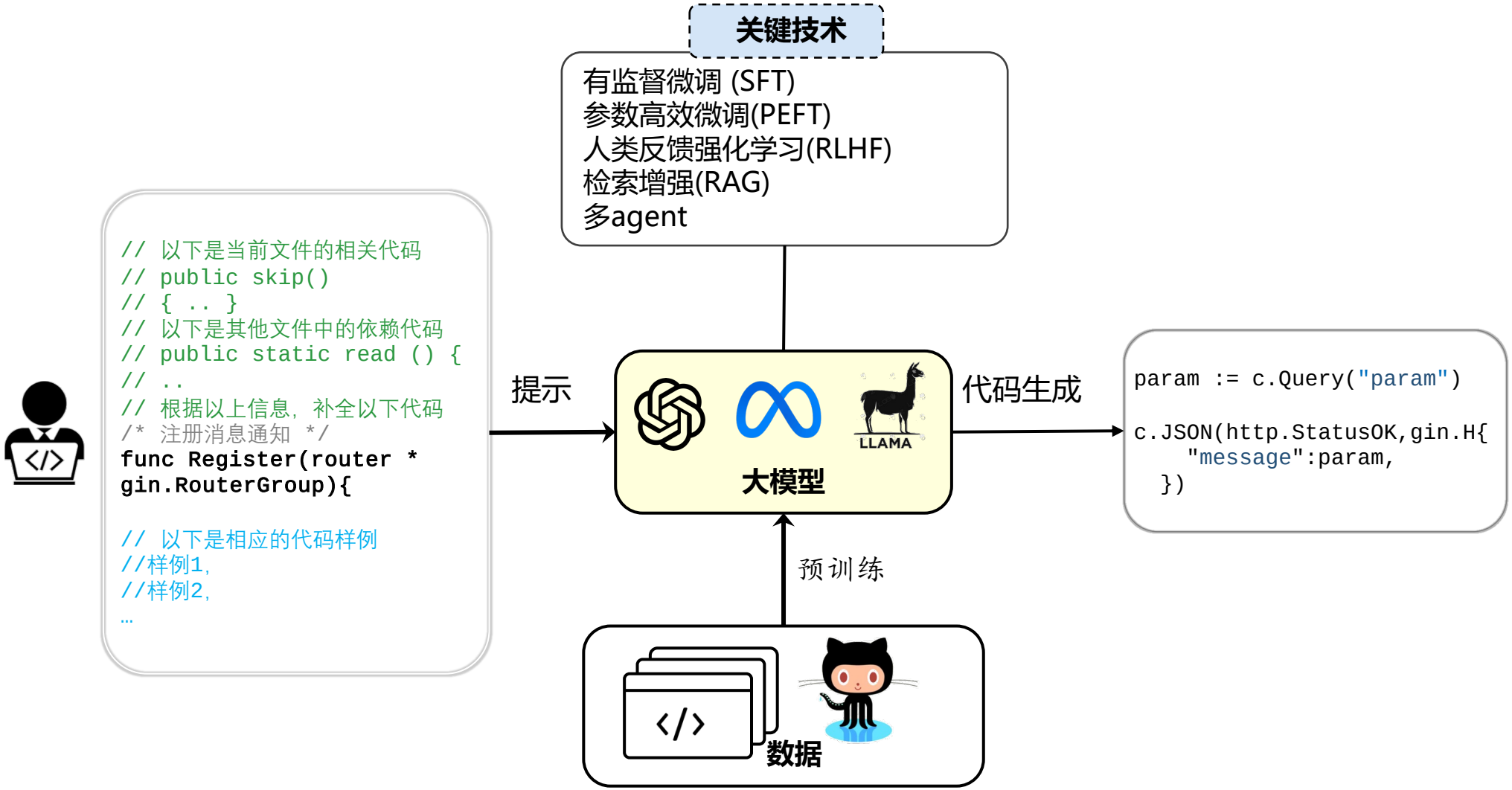
- 跨语言代码搜索
- 搜索语句重构

目录
CONTENTS

PART 01

背景：程序生成问题与痛点

基于大模型的程序自动生成



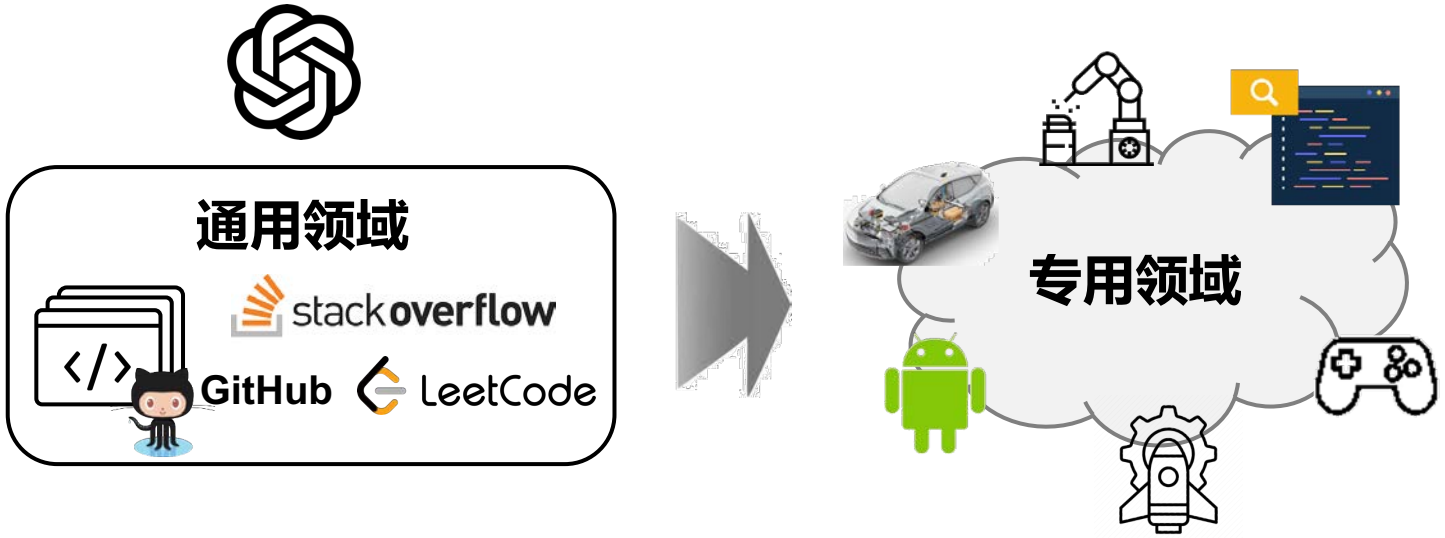
研究现状

HumanEval评测榜单

Rank	Model	Pass@1↑	Pass@10	Pass@100	Extra Training Data	Paper	Code	Result	Year
1	**Claude 3.5 Sonnet** (0-shot)	92.0			×	Claude 3.5 Sonnet Model Card Addendum			2024
2	**GPT-4o** (0-shot)	90.2			×	Claude 3.5 Sonnet Model Card Addendum			2024
3	**GPT-4T** (0-shot)	87.1			×	Claude 3.5 Sonnet Model Card Addendum			2024
4	**OctorCoder** (GPT-4)	86.6			×	OctoPack: Instruction Tuning Code Large Language Models			2023
5	**ANPL** (GPT-4)	86.6			×	ANPL: Towards Natural Programming with Interactive Decomposition			2023
6	**MetaGPT** (GPT-4)	85.9			×	MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework			2023
7	**Parsel** (GPT-4 + CodeT)	85.1			×	Parsel: Algorithmic Reasoning with Language Models by Composing Decompositions			2022
8	**Claude 3 Opus** (0-shot)	84.9			×	The Claude 3 Model Family: Opus, Sonnet, Haiku			2024

挑战

大模型的领域面广而不深



- 大模型依赖海量的训练数据，而专用领域的训练数据有限
- 大模型在开放领域数据上预训练，缺少专用领域的知识（API、业务知识、项目私有函数等等）

PART 02

探究：大模型在业务领域代码生成方面的表现

经验研究

RQ1: 大模型在特定业务领域的代码生成能力如何?

RQ2: 如何提示大模型生成特定业务领域的代码?

RQ3: 如何将领域知识引入代码大模型?

Gu X, Chen M, Lin Y. et al. On the effectiveness of Large Language Models in Domain Specific Code Generation. TOSEM 2024

RQ1:大模型在特定业务领域上的代码生成能力

- **测试模型**：ChatGPT, CodeLlama-7B, PolyCoder-2.7B
- **测试方法**：根据函数签名补全函数体
- **测试指标**：BLEU, CodeBLEU, 人工评估
- **测试数据**：
  - 开放领域数据集：HumanEval, CodeSearchNet (Go)
  - 专用领域数据集：GitHub上使用了特定领域(web, game)第三方库的500个函数

Library	Language	Domain	Description	Popularity	# train functions
Gin	Go	web development	web framework featuring Martini-like APIs	69.5k stars	21,161
gRPC-go	Go	web development	remote procedure call across data centers	18.3k stars	129,803
Prometheus	Go	web development	client library for application instrumentation	9.1k stars	11,708
Unreal Engine	C++	game development	realtime 3D game engine	-	184,808
Cocos2d-x	C++	game development	cross-platform 2D game framework	17.2k stars	27,119
Bgfx	C++	game development	cross-platform rendering library	13.1k stars	12,285

测试结果

- **定量分析**

Dataset	General-Purpose		Code-Oriented			
	ChatGPT-3.5-154B		*CodeLlama-7B*		*PolyCoder-2.7B*	
	BLEU	**CodeBLEU**	**BLEU**	**CodeBLEU**	**BLEU**	**CodeBLEU**
Open-domain datasets						
HumanEval	39.00	30.05	18.13	17.99	22.22	13.81
CodeSearchNet (Golang)	11.16	25.29	16.14	19.63	15.97	19.56
Domain-specific datasets						
Gin	5.81	17.73	8.47	18.07	8.13	17.05
Prometheus	1.75	12.17	1.79	13.13	1.77	12.11
gRPC-go	2.53	12.15	57.61	60.39	55.36	57.52
Unreal Engine	5.22	10.57	0.73	10.21	0.94	9.87
cocos2d-x	3.71	12.92	16.76	25.87	13.83	23.83
bgfx	0.86	8.09	0.55	7.72	0.76	7.16

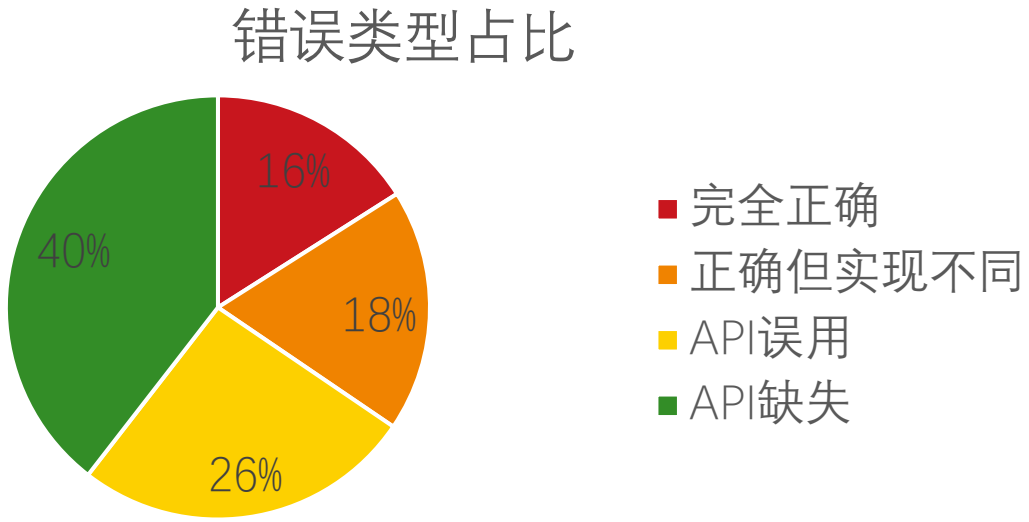
发现1: 大模型在特定领域上的代码生成效果有明显下降。

发现2: ChatGPT(开放域大模型)在开放域数据集上效果优于代码专用的大模型。

测试结果

- **定性分析**

  随机抽取200个样例进行人工分析



发现3：大模型在专用领域的代码生成效果下降，主要因为缺少第三方库的知识，如API的使用方式

测试结果

- **常见错误：**
  - **功能错误**：生成的代码功能不符合用户需求，或者缺失部分功能
  - **函数调用错误**：生成的代码语义大致正确，但没有调用正确的API

举例：

Reference code:

```
func (api *API) RegisterRouter(router *gin.RouterGroup){
    v1 := router.Group(constants.APIVersion1)
    api.execute.Register(v1)
}
```

ChatGPT:

```
func (api *API) RegisterRouter(router *gin.RouterGroup){
    router.GET("/test", api.Test)
}
```

RQ2: 如何提示大模型生成领域特定代码

- **测试方法**：在大模型提示中加入知识，进行(零样本)代码生成
- **提示模板:** 基础信息(函数签名、包引入、API等) 及其组合

Prompt type	Example
basic prompts	
Function signature	`Complete this function: func Routes(r *gin.Engine)`
Library import	`Complete this function using gin: func Routes(r *gin.Engine)`
API	`Complete this function using gin.RouterGroup.Use: func Routes(r *gin.Engine)`
docstring	`gin.RouterGroup.Use adds middleware to the group.` `Complete this function: func Routes(r *gin.Engine)`
combined prompts	
API + docstring	`Complete this function using gin.RouterGroup.Use.` `gin.RouterGroup.Use adds middleware to the group. func Routes(r *gin.Engine)`
docstring + API	`gin.RouterGroup.Use adds middleware to the group.` `Complete this function using gin.RouterGroup.Use: func Routes(r *gin.Engine)`

测试方案

提示模板

- 1. 不使用提示，仅使用函数签名，作为对照组

```
func Login(c *gin.Context){
```

- 2. Import语句作为提示

```
import{  "github.com/gin-gonic/gin"  }
func Login(c *gin.Context){
```

- 3. API序列作为提示

```
// GetApp BindArgsWithGin APIError GetUserByAccount APIError ...
func Login(c *gin.Context){
```

- 4. API知识作为提示(即API文档)

```
// DefaultQuery : DefaultQuery returns the keyed url query value if it exists ...
// MustGet : MustGet returns the value for the given key if it exists ...
func ValidatePipeline(c *gin.Context) {
```

测试结果

- 各种提示类型和组合在特定领域代码生成上的效果对比（ChatGPT）

Prompt type	Gin		Prometheus		gRPC-go		Unreal Engine		Cocos2d-x		bgfx	
	BLEU	**CB**	**BLEU**	**CB**	**BLEU**	**CB**	**BLEU**	**CB**	**BLEU**	**CB**	**BLEU**	**CB**
function signature	5.81	17.73	1.75	12.17	2.53	12.15	5.22	10.57	3.71	12.92	0.86	8.09
+ library import	6.08	17.83	3.37	15.12	5.44	16.93	5.79	11.27	5.37	14.68	1.02	8.79
+ API	**14.86**	26.69	**7.18**	20.51	**22.24**	**31.84**	**14.78**	18.34	**14.18**	**22.35**	**1.39**	11.32
+ docstring	14.56	**27.04**	7.04	**20.78**	22.07	31.74	14.58	**18.35**	14.06	**22.35**	1.33	**11.40**
+ docstring	5.57	17.78	1.94	12.31	2.54	12.33	5.32	10.41	3.57	12.93	0.88	8.38
+ API	13.64	25.94	6.95	20.48	22.08	31.64	14.12	17.74	14.00	22.24	1.23	11.03

发现4：将第三方库知识通过提示引入可以提升大模型在特定领域的代码生成能力

PART 02

方法：业务知识如何融入大模型？

业务知识如何有效的融入大模型?

总体思路：自动为大模型生成知识提示，并引导大模型生成特定领域代码

关键问题：
1.领域知识从哪里来?
2.如何将提示融入代码生成的过程?

Gu X, Chen M, Lin Y. et al. On the effectiveness of Large Language Models in Domain Specific Code Generation. TOSEM 2024

领域知识从哪里来?

- **外部知识**：API文档

API

功能概括

用法描述

Incubator-brpc文档示例

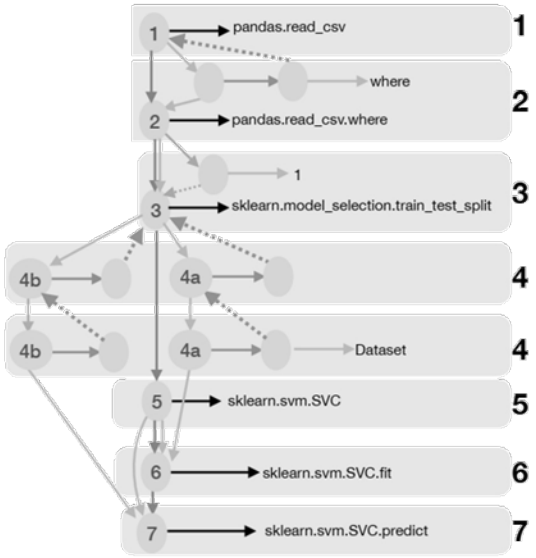
parse

```
typedef ParseResult (*Parse)(butil::IOBuf* source, Socket *socket, bool re
```

This function is used to cut messages from source. Client side and server side must share the same parse function. The returned message will be passed to `process_request` (server side) or `process_response` (client side).

- **内部知识**：代码结构
  - 函数间的调用关系
  - 函数（API）的调用顺序

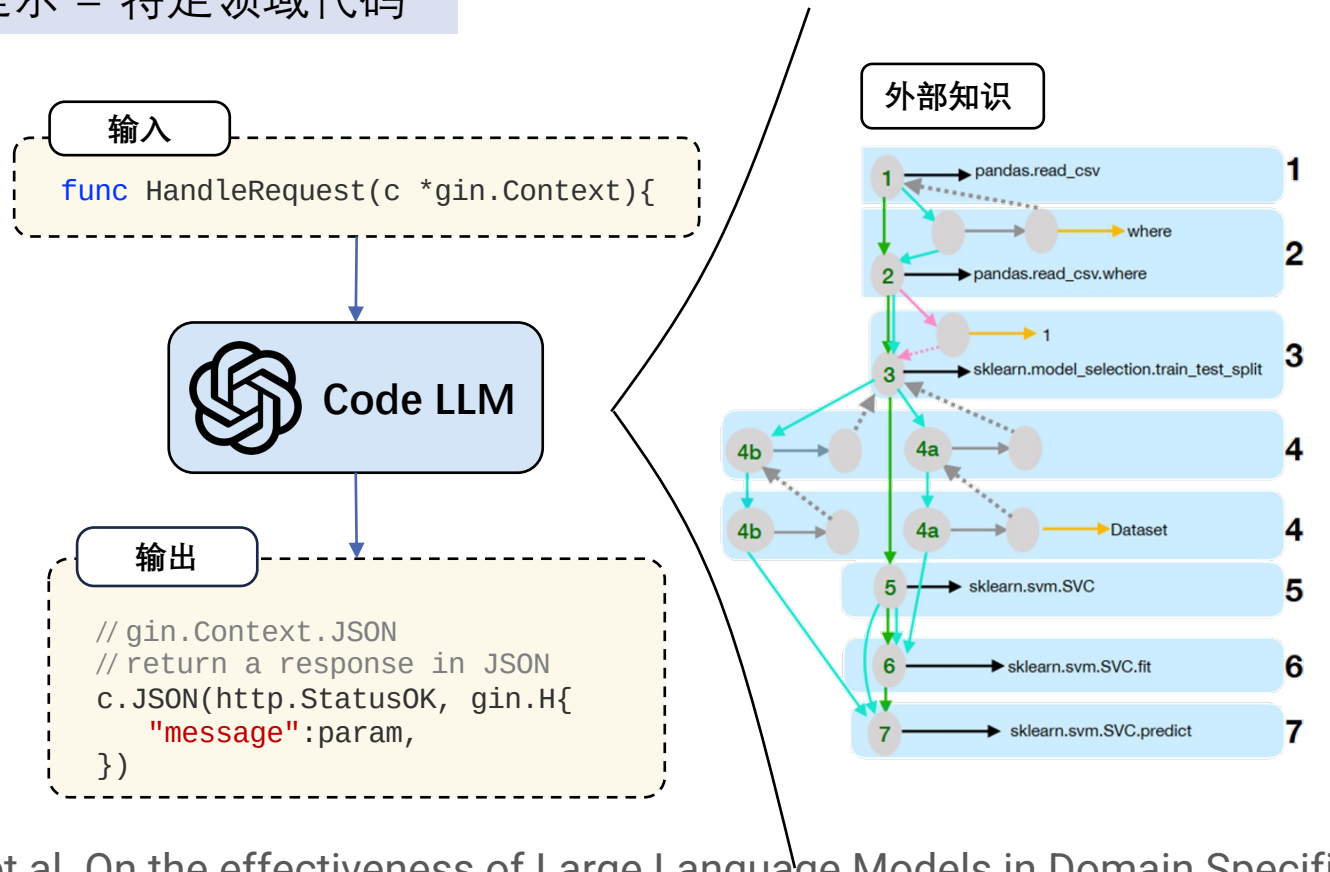
  (业务流程与逻辑)



如何将提示融入代码生成的过程?

- **DomCoder: 领域代码生成新方法**

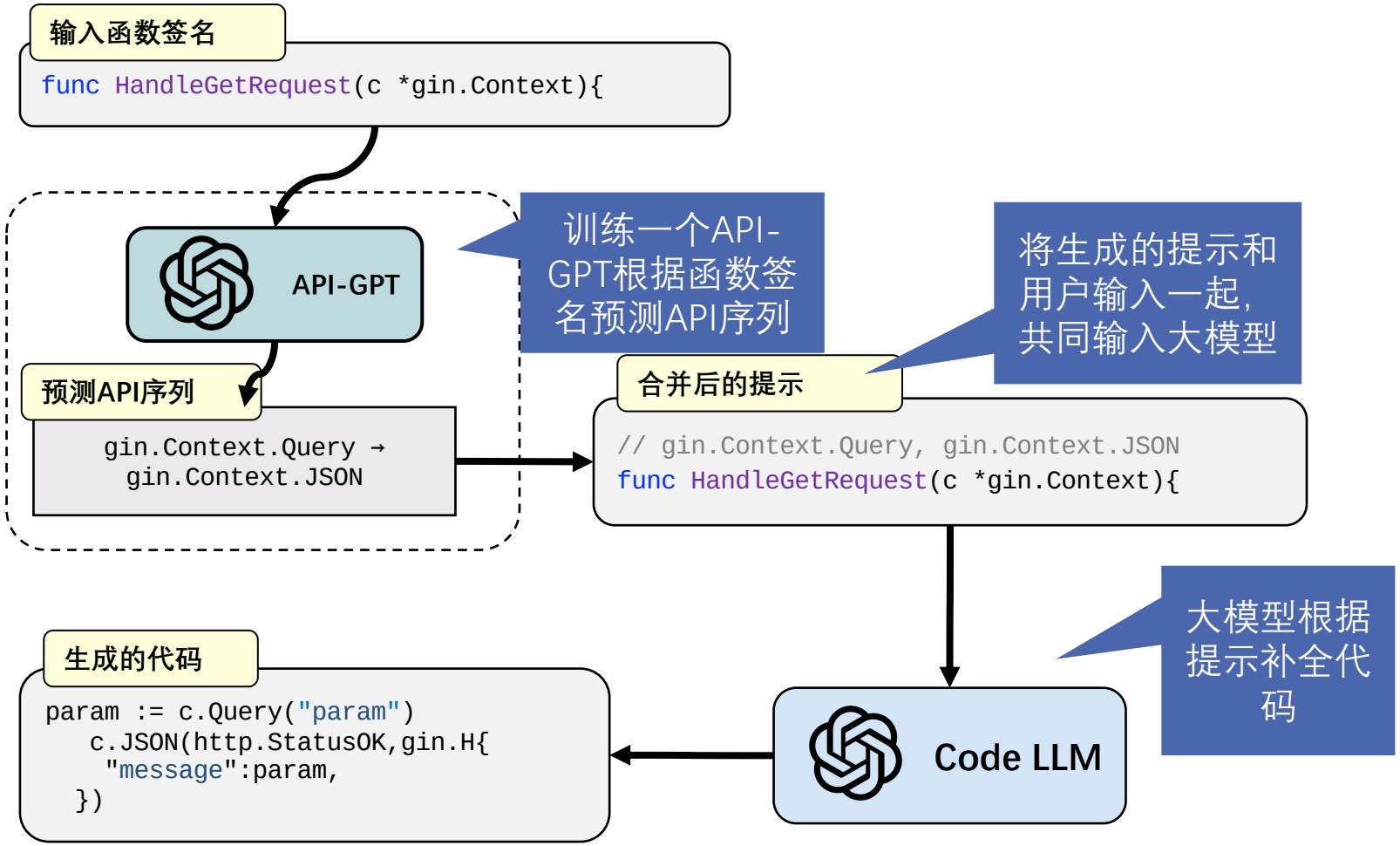
API知识 + 提示 = 特定领域代码



Gu X, Chen M, Lin Y. et al. On the effectiveness of Large Language Models in Domain Specific Code Generation. TOSEM 2024

策略一：API知识外挂

- 收集特定项目的函数调用序列，训练外挂API-GPT，作为API提示辅助

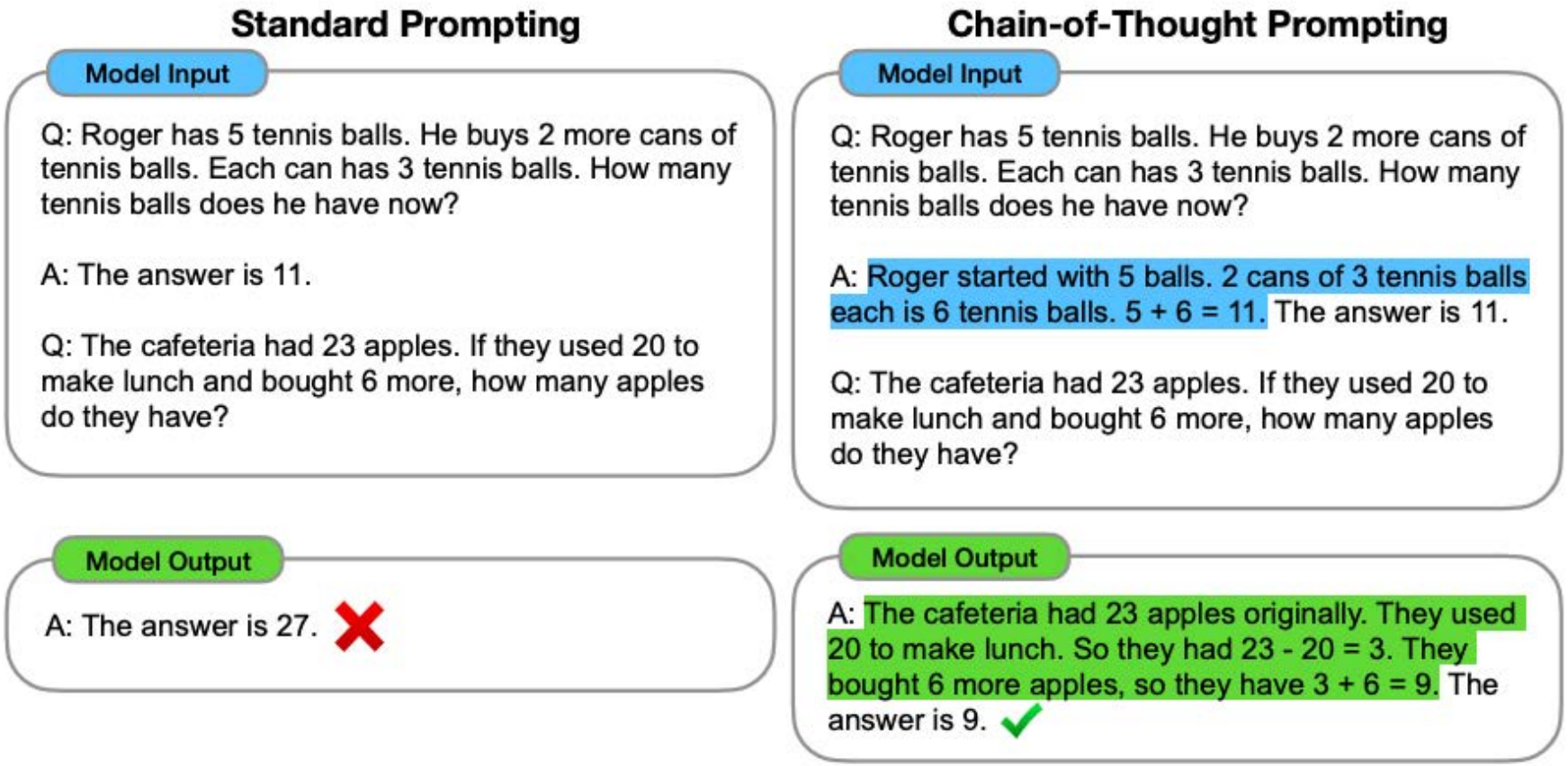


举例

- 输入函数签名：
```
func ValidatePipeline(c *gin.Context) {
```
- 借助另外训练的知识引擎（API-GPT、API-Bot）生成外部知识（如API调用序列）：
```
// call sequence: MustGet Retrieve Retrieve Retrieve Retrieve GetFullName ...
```
- 从外部知识库中查询相关的API知识:
```
// DefaultQuery : DefaultQuery returns the keyed url query value if it exists ...
// MustGet : MustGet returns the value for the given key if it exists ...
```
- 组合起来，作为给模型的提示：
```
// call sequence: MustGet Retrieve Retrieve Retrieve Retrieve GetFullName ...
// DefaultQuery : DefaultQuery returns the keyed url query value if it exists ...
// MustGet : MustGet returns the value for the given key if it exists ...
func ValidatePipeline(c *gin.Context) {
```
- 模型预测出代码：
```
func ValidatePipeline(c *gin.Context) {
    m := c.MustGet("metadata").(*types.Metadata)
    ...
```

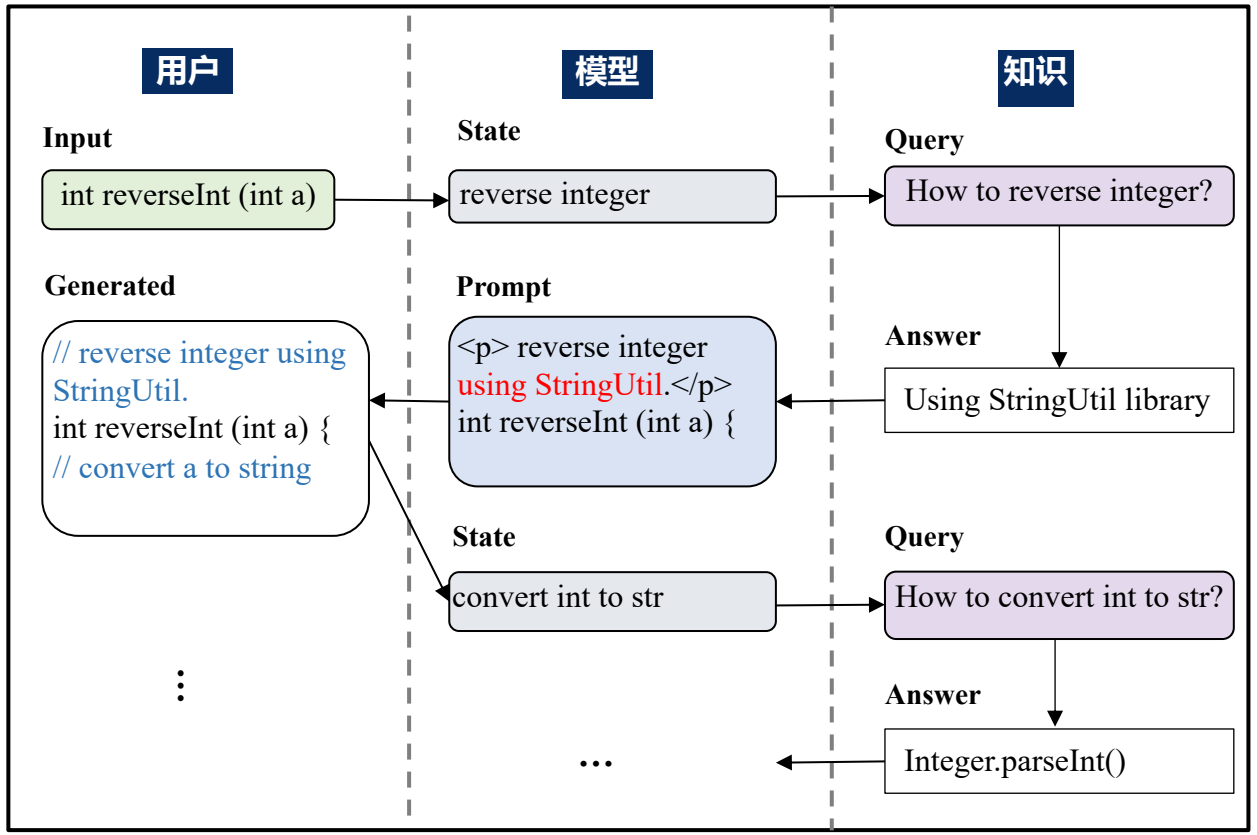
思维链提示

- **思维链**：不直接生成最终答案，而是提示大模型输出答案的思路，逐步引导答案。



从思维链看代码生成过程

- 将代码生成看成多步的“解题过程”，在代码生成过程中同步进行问题思考和知识提示。



初始输入

code

```
func HandleRequest(c *gin.Context){
```

Step 0

API state
task state
code

```
// gin.Context.Query
// get the request parameter
param := c.Query("param")
```

Step 1..n-1

... ...

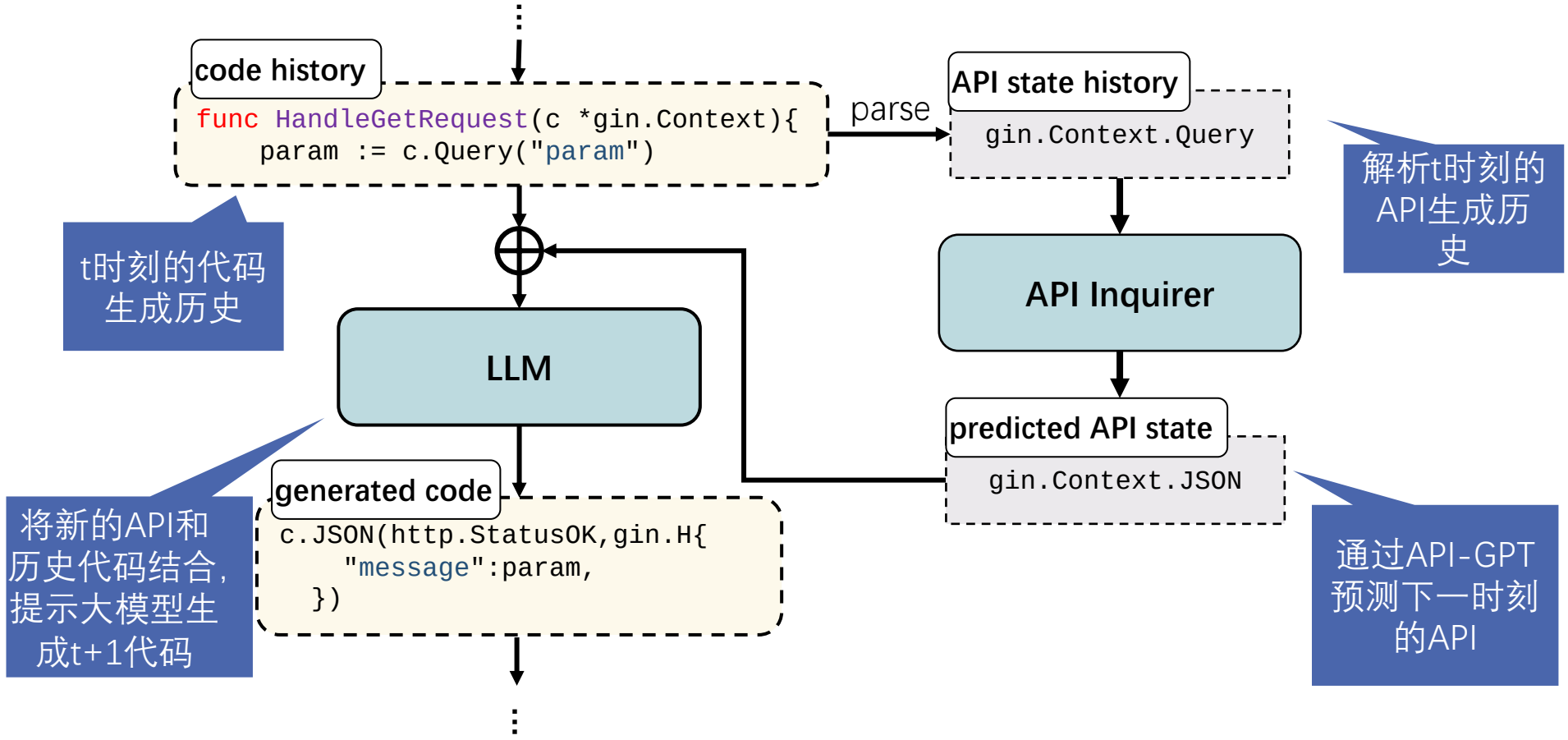
Step n

API state
task state
code

```
// gin.Context.JSON
// return a response in JSON
c.JSON(http.StatusOK, gin.H{
   "message":param,
})
```

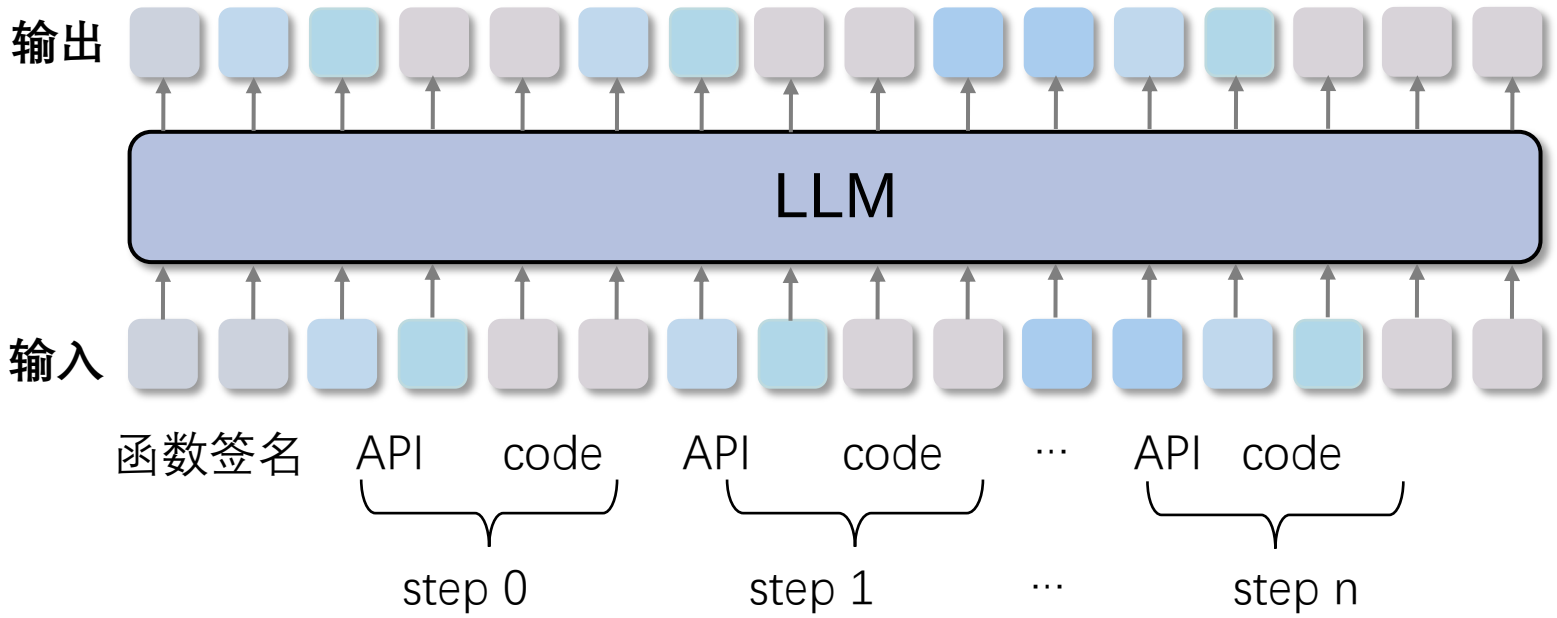
策略二：思维链提示(CoT-PT)

- 借助额外的知识生成模型，如API-GPT，但采用思维链逐步提示大模型



策略三：思维链微调（CoT-FT）

- 不借助外挂知识，将代码生成和API知识思维(Thought)统一为一个序列生成问题
- 微调同一个大模型， 进行**一体化端到端**生成



实验结果

- **大模型**: PolyCoder (GPT-2, 2.7B)、StarCoder-15.5B、CodeLlama-7B
- 分别在zero-shot和SFT设定下和原始LLM进行对比

Model	gin		prometh.		grpc-go		UnrealEng.		cocos2d-x		bgfx	
	BLEU	**CB**	**BLEU**	**CB**	**BLEU**	**CB**	**BLEU**	**CB**	**BLEU**	**CB**	**BLEU**	**CB**
PolyCoder (zero-shot)	8.13	17.05	1.77	12.11	55.36	57.52	0.94	9.87	13.83	23.83	0.76	7.16
+In-context learning [1]	8.18	16.36	2.01	12.57	55.54	57.11	0.96	0.98	13.27	23.00	0.83	7.24
+*DomCoder* (kg-GPT)	**9.91**	18.16	**2.29**	12.76	**56.87**	**58.30**	1.26	**10.20**	**15.33**	**24.09**	**0.78**	7.34
+*DomCoder* (CoT-PT)	9.17	17.41	2.21	**12.65**	55.97	57.70	**1.38**	10.15	16.13	24.91	**0.91**	**7.87**
PolyCoder (fine-tuning)	20.62	27.18	5.88	17.11	62.97	63.37	1.57	11.10	27.65	32.67	1.01	11.80
+*DomCoder* (CoT-FT)	**21.12**	**27.51**	**7.59**	**19.13**	**63.23**	**63.83**	**2.26**	**11.69**	**29.30**	**33.99**	**1.05**	**11.95**
StarCoder (zero-shot)	8.46	17.55	1.68	12.37	58.69	60.09	0.58	9.00	14.08	23.56	0.80	7.21
+In-context learning [1]	7.11	17.32	2.00	12.75	58.12	59.07	1.14	10.76	18.62	26.55	1.12	7.93
+*DomCoder* (kg-GPT)	**13.12**	**20.44**	**3.75**	14.74	**60.41**	**61.85**	**1.74**	**10.98**	**22.43**	**27.53**	**1.20**	**8.67**
+*DomCoder* (CoT-PT)	12.46	19.77	3.32	13.66	59.91	61.25	1.64	10.96	20.78	25.81	1.17	8.42
CodeLlama (zero-shot)	8.47	18.07	1.79	13.13	57.61	60.39	0.73	10.21	16.76	25.87	0.55	7.72
+In-context learning [1]	8.22	18.58	2.69	13.87	**59.35**	**61.49**	0.81	10.35	17.90	27.06	0.60	8.47
+*DomCoder* (kg-GPT)	11.71	19.83	2.62	13.41	55.93	58.24	0.99	10.55	**19.58**	**27.71**	**0.65**	**8.66**
+*DomCoder* (CoT-PT)	**11.75**	**19.90**	**3.07**	**14.58**	56.03	58.69	**1.15**	**11.15**	16.39	25.16	0.57	8.21

引入第三方库知识提升大模型在特定领域的代码生成能力

实验结果

- **行业研究**
  - 腾讯tRPC-go相关代码上测试CodeLlama-7B
  - 人工评估相关性和正确性

Model	Task Relevance	Semantic Correctness
CodeLlama-7B (zero-shot)	20/30	6/30
- w/ *DomCoder* (kg-GPT)	28/30	18/30

Gu X, Chen M, Lin Y. et al. On the effectiveness of Large Language Models in Domain Specific Code Generation. TOSEM 2024

实验结果

• 实例分析

Reference Code

```
func ensureNotLoggedIn() gin.HandlerFunc {
    return func(c *gin.Context) {
        loggedInInterface, _ := c.Get("is_logged_in")
        loggedIn := loggedInInterface.(bool)
        if loggedIn {
            c.AbortWithStatus(http.StatusUnauthorized)
        }
    }
}
```

PolyCoder (zero-shot)

```
func ensureNotLoggedIn() gin.HandlerFunc {
    return func(c *gin.Context) {
       if err := ensureLoggedIn(c); err!= nil {
           internal.HandleError(err, func() {
             c.AbortWithError(http.StatusUnauthorized,err)
           })
        }
}
```

PolyCoder + kgGPT

```
// gin.Context.AbortWithStatusJSON, gin.Context.GetHeader
func ensureNotLoggedIn() gin.HandlerFunc {
    return func(c *gin.Context) {
        if err := ensureLoggedIn(c); err!= nil {
            c.AbortWithStatusJSON(http.StatusForbidden,
                    err.Error())
             return
        }
}
```

API knowledge

PolyCoder + CoT-PT

```
func ensureNotLoggedIn() gin.HandlerFunc {
    return func(c *gin.Context) {
        // gin.Context.Request.Context
        ctx := c.Request.Context()
        if ctx.Get(userContextKey) == nil {
            ctx = ctx.WithField(userContextKey,
              auth.SessionID(req.FormValue(userKey)))
        }
        // gin.Context.JSON
        gin.Context.JSON(c.Writer, c.Request,
                         auth.AuthRequestWrapper){
            Status:  auth.OK,
            Method:  auth.MethodGet,
        }
    }
}
```

API state

API state

PolyCoder-FT

```
func ensureNotLoggedIn() gin.HandlerFunc {
    return func(c *gin.Context) {
        if valid := c.Errors.Form()!= nil {
           DataToValidErrors(c, errors.New(“empty integer”))
           return
        }
        leader := GB(c)
        if err := c.ShouldBindJSON(&remove); err!=nil{
            warnErr(c, err)
            c.IndentedJSON(http.StatusBadRequest,
                           gin.H{"error": err.Error()})
            return
        }
}
```

PolyCoder + CoT-FT

```
func ensureNotLoggedIn() gin.HandlerFunc {
    return func(c *gin.Context) {
        // Get the value for the given key
        // gin.Context.Get
        loggedInInterface, _ := c.Get("is_logged_in")
        loggedIn := loggedInInterface.(bool)
        if !loggedIn {
            // AbortWithStatus calls `Abort()`
            // and writes the headers
            // gin.Context.AbortWithStatus
            c.AbortWithStatus(http.StatusUnauthorized)
        }
    }
}
```

Task state

API state

Task state

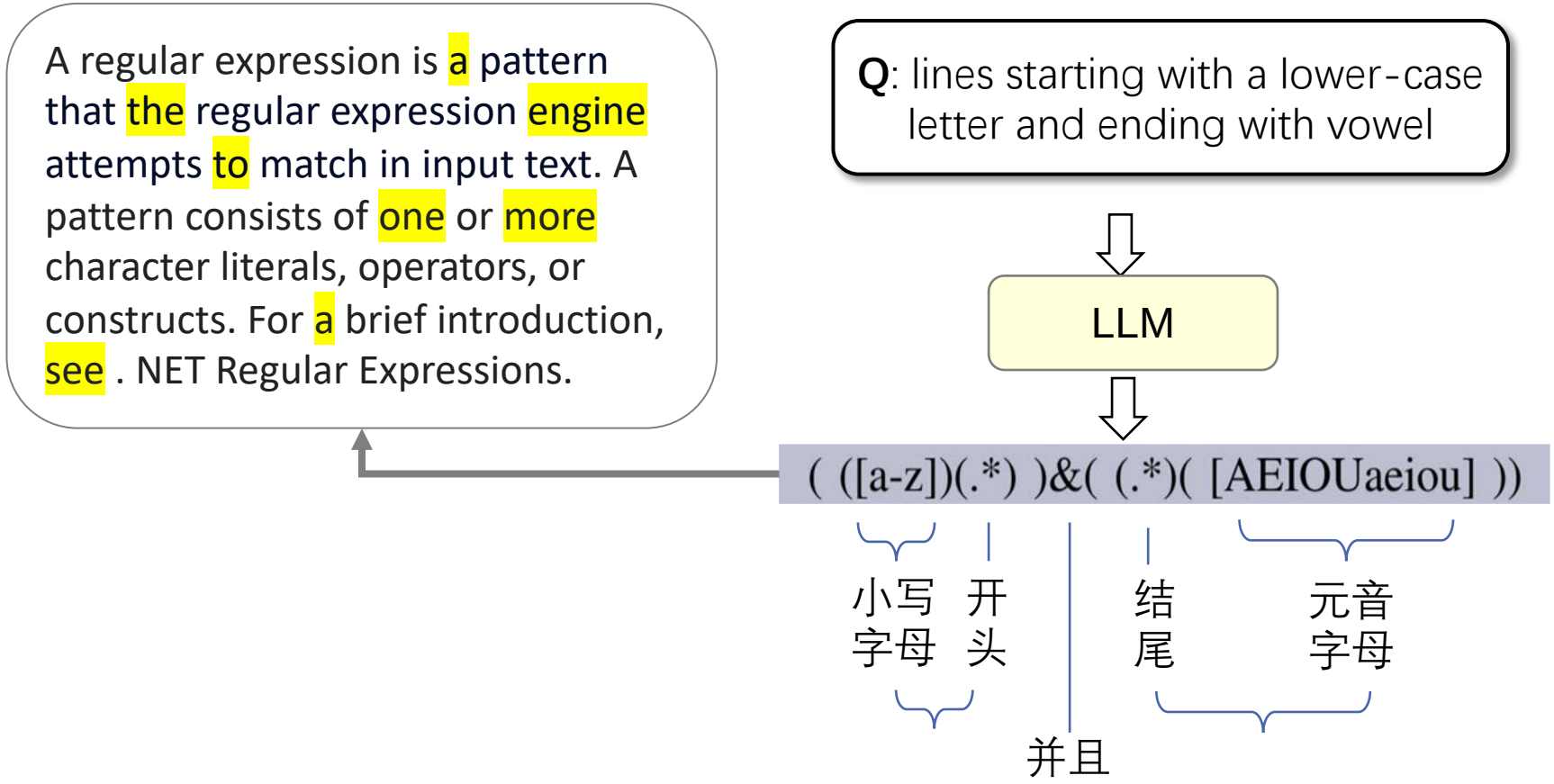
API state

PART 03

深入：基于思维链的正则表达式生成

正则表达式自动生成

- **正则表达式**：对字符串操作的一种逻辑公式，用特定字符及其组合组成一个“规则字符串”，用来表达对字符串的一种过滤逻辑。

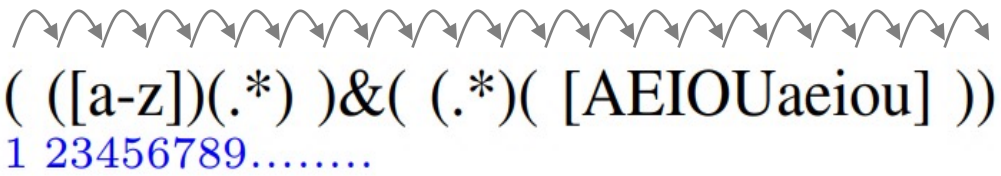


基于深度学习的正则表达式自动生成

- **面临问题**：传统的**自回归**生成方式无法反映真实的字符匹配顺序。

Q: lines starting with a lower-case letter and ending with vowel

自回归语言模型的生成顺序：



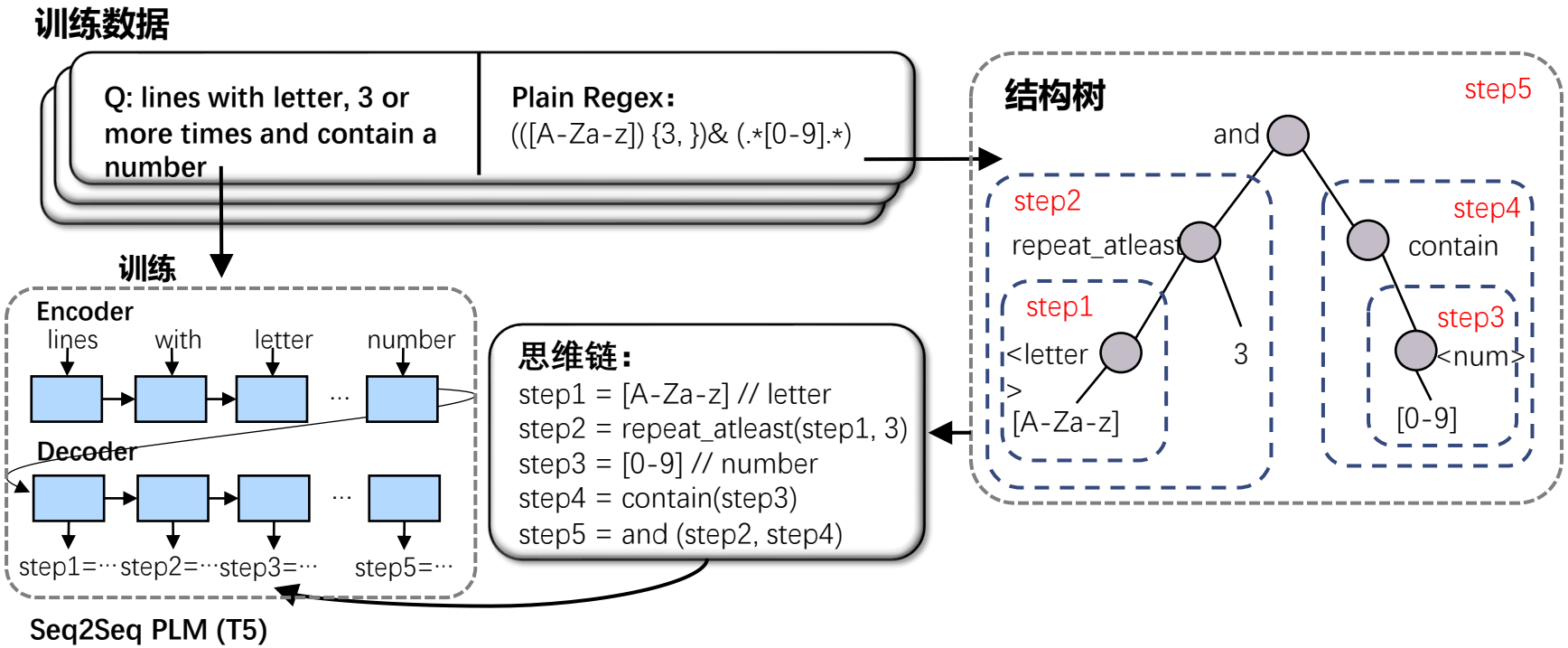
中间步骤

步骤 1	lowercase	[a-z]
步骤 2	start with	[a-z](.*)
步骤 3	vowel	[AEIOUaeiou]
步骤 4	end with	(.*)[AEIOUaeiou]
步骤 5	and	([a-z](.*))&((.*)([AEIOUaeiou]))

实际的文本匹配顺序

Idea：生成思维链而不是代码？

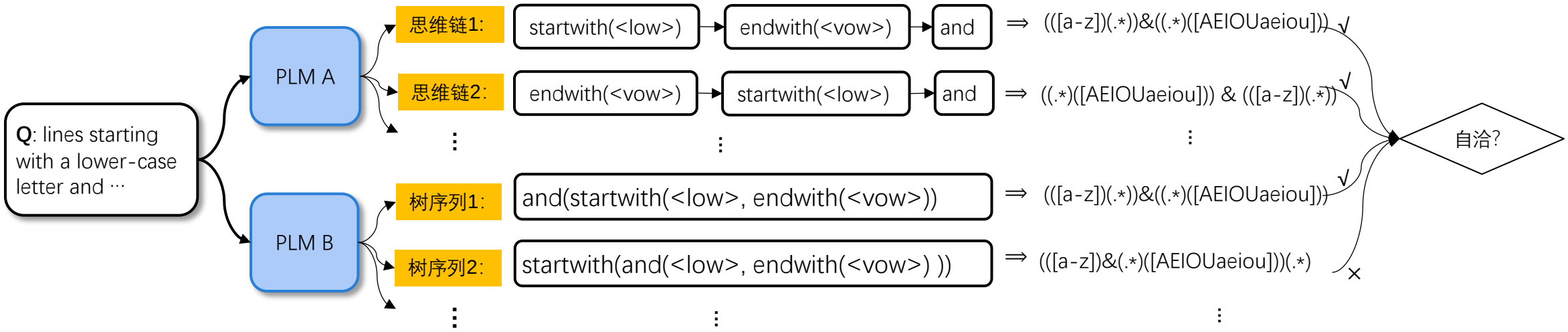
- 将正则表达式分解成思维链
- 训练<自然语言，思维链>数据对



Shuai Zhang, Xiaodong Gu, Yuting Chen, Beijun Shen. InfeRE: Step-by-Step Regex Generation via Chain of Inference. ASE 2023

基于思维链的正则表达式生成

- 采用多个思维链的逻辑**自洽性**提高结果的可靠程度



实验设置

- **数据集：**

数据集	训练	验证	测试
NL-RX-Turk	6,500	1,000	2,000
KB13	618	206	206

- **精度指标：**
  - **DFA-EQ**: 将正则表达式转换成确定有穷自动机，评估它们的语义等价性
  - **EM**: 生成完全一样表达式的比例.

实验结果

Approach	NL-RX-Turk			KB13		
	DFA-EQ@1(%)	DFA-EQ@5(%)	EM(%)	DFA-EQ@1(%)	DFA-EQ@5(%)	EM(%)
Semantic-Unify	38.6	—	—	65.5	—	—
Deep-Regex$^{\mathrm{MLE}}$	60.3	76.0	40.7	66.5	75.7	55.8
Deep-Regex$^{\mathrm{MML}}$	62.4	76.8	39.2	68.2	77.7	56.8
SemRegex	62.3	—	—	78.2	—	—
SoftRegex	62.8	72.1	41.5	78.2	79.6	62.1
S_2RE	62.8	—	—	78.2	—	—
S_2RE-T5	67.6	85.7	54.4	82.0	88.8	71.4
TRANX	58.8	75.6	44.0	73.8	82.0	61.2
InfeRE (ours)	**69.2**	**89.3**	53.4	**82.5**	**91.3**	69.4
- w/o SC	67.8	85.9	**55.5**	81.6	87.9	**72.3**
- w/o SC+CI	67.2	85.5	54.2	82.0	88.8	70.4

采用思维链和自洽机制对程序进行分解可以有效提高正则表达式生成的准确度

实验结果

- **结果示例**

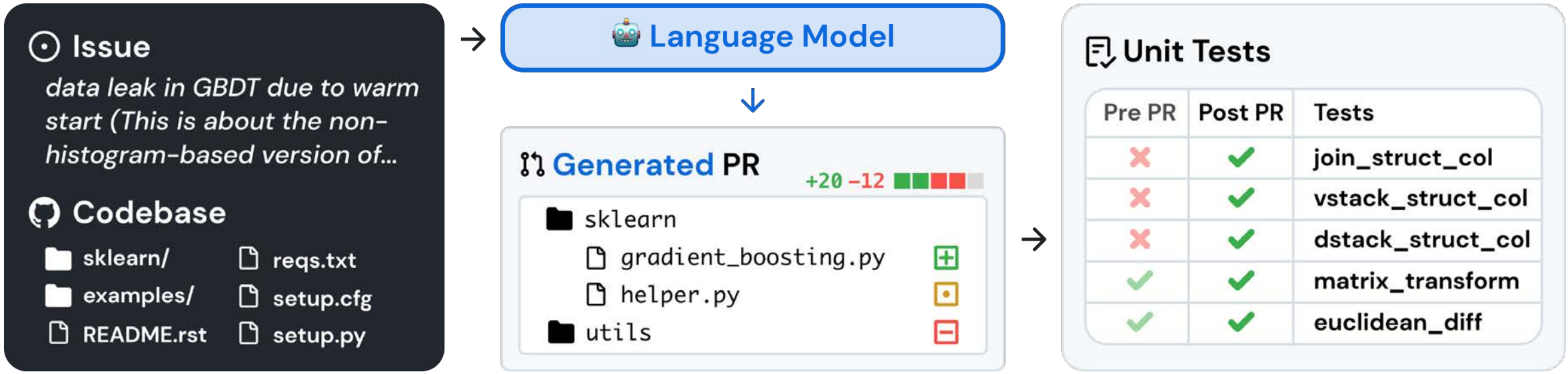
Example 1.	Query: *either start with string <M0> or end with any number.*	
Gold:	**((.*)([0-9]))—((<m0>)(.*))**	**or(endwith(<num>),startwith(<m0>))**
Deep-Regex^{MML}:	((<m0>)—((.*)([0-9])))(.*)	startwith(or(<m0>,endwith(<num>)))
SoftRegex:	((<m0>)—(.*)([0-9]))(.*)	startwith(or(<m0>,endwith(<num>)))
InfeRE-w/o SC+CI:	((<m0>)—((.*)([0-9])))(.*)	startwith(or(<m0>,endwith(<num>)))
InfeRE-w/o SC:	((<m0>)(.*))—((.*)([0-9]))	or(startwith(<m0>),endwith(<num>))
InfeRE:	((<m0>)(.*))—((.*)([0-9]))	or(startwith(<m0>),endwith(<num>))
Example 2.	Query: *lines with the string <M0> ending in zero or more of a capital letter.*	
Gold:	**(<m0>)(((.*)([<A-Z>]))*)**	**concat(<m0>,star(endwith(<cap>)))**
Deep-Regex^{MML}:	((<m0>)(1,))—((.*)([A-Z]))	or(repeat_least(<m0>,1),endwith(<cap>))
SoftRegex:	((<m0>)(1,))&((.*)([A-Z]))	and(repeat_least(<m0>,1),endwith(<cap>))
InfeRE-w/o SC+CI:	(.*(.*<m0>.*))*	star(endwith(contain(<m0>)))
InfeRE-w/o SC:	(<m0>)(((.*)([<A-Z>]))*)	concat(<m0>,star(endwith(<cap>)))
InfeRE:	(<m0>)(((.*)([<A-Z>]))*)	concat(<m0>,star(endwith(<cap>)))
Example 3.	Query: *items with a any letter preceding <M0>.*	
Gold:	**(([A-Z])—([a-z]))(<m0>)**	**concat(or(<low>,<cap>),<m0>)**
Deep-Regex^{MML}:	(([a-z])&([a-z]))(<m0>)	concat(and(<low>,<low>),<m0>)
SoftRegex:	([a-z])(<m0>)	concat(<low>,<m0>)
InfeRE-w/o SC+CI:	(([A-Z])&([a-z]))(<m0>)	concat(and(<low>,<cap>),<m0>)
InfeRE-w/o SC:	(([A-Z])&(([a-z])&(a-zA-Z)))(<m0>)	concat(and(and(<low>,<cap>),<let>),<m0>)
InfeRE:	(([A-Z])—(([a-z])—(a-zA-Z)))(<m0>)	concat(or(or(<low>,<cap>),<let>),<m0>)

PART 04

实践：复杂程序问题修复

代码修复：程序生成的最后一公里

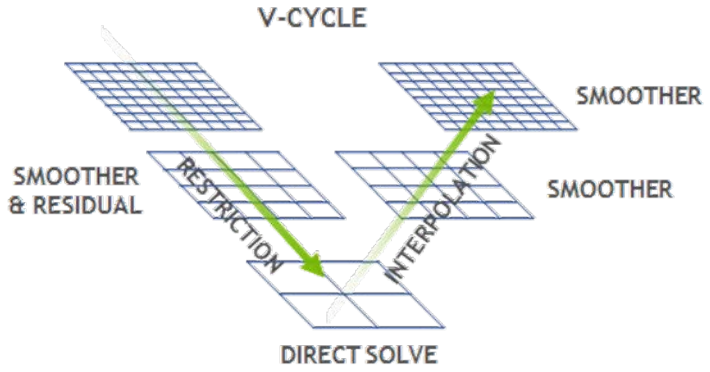
- 软件开发 = 大模型生成代码+人工修复代码



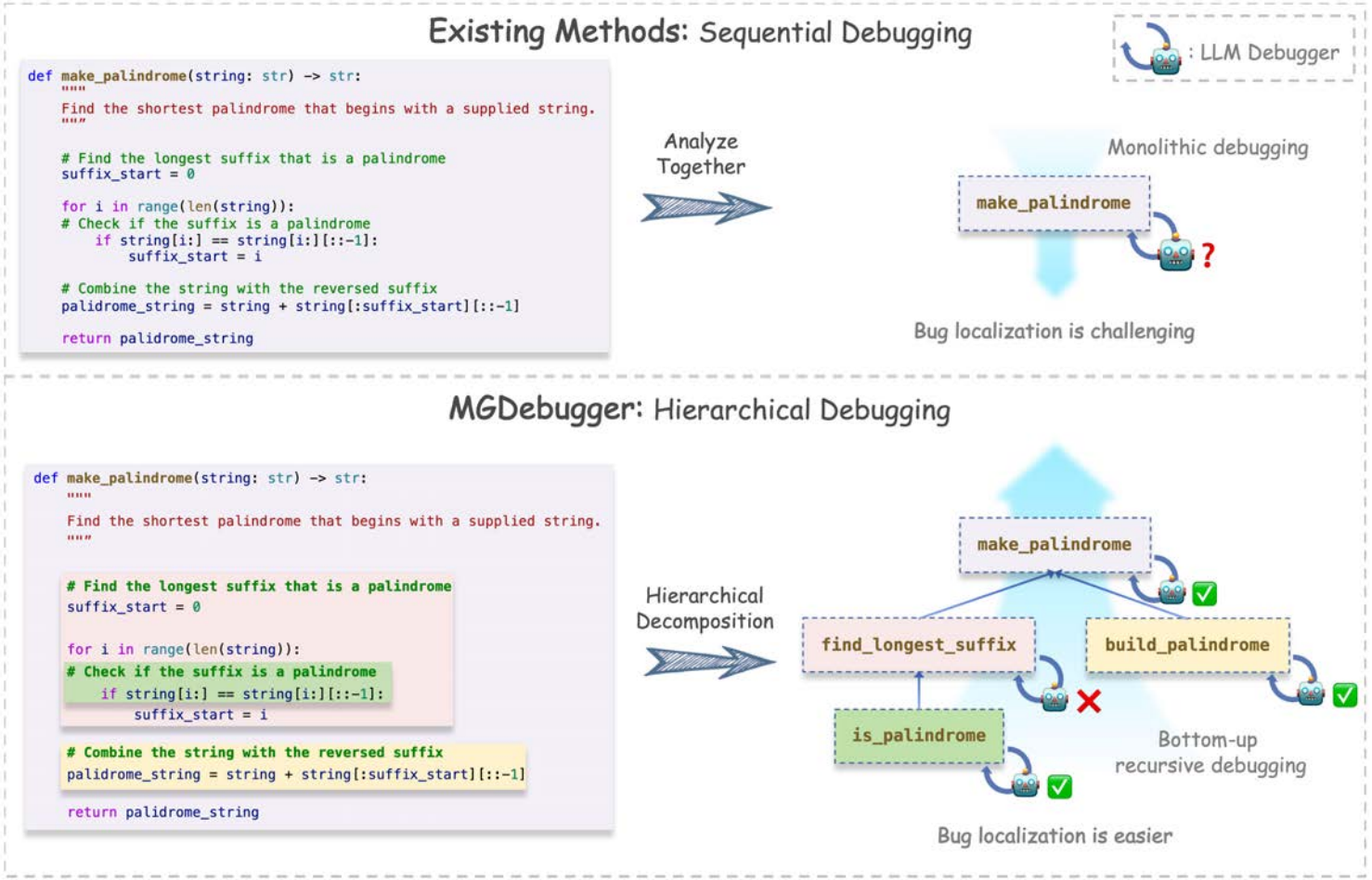
如何让大模型辅助修复复杂(仓库级)代码？如何让大模型持续学习仓库级代码知识?

MGDebugger: 多粒度程序修复

- **多粒度程序修复**



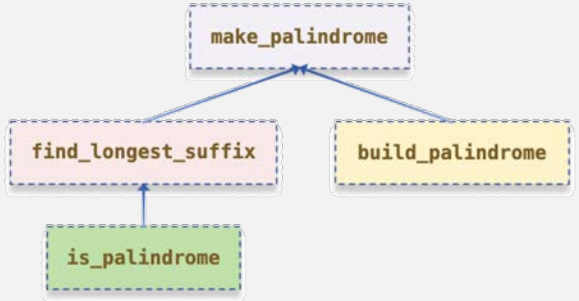
把程序错误分成多个层级：算法层→模块层→语法层→…



Yuling Shi, Songsong Wang, Chengcheng Wan, Xiaodong Gu. From Code to Correctness: Closing the Last Mile of Code Generation with Hierarchical Debugging [Arxiv 2024]

MGDebugger: 多粒度程序修复

① 任务拆解



② 子模块测试用例生成

```
assert find_longest_suffix('cat')    == 2  # Got 2
assert find_longest_suffix('cata')   == 1  # Got 3
assert find_longest_suffix('')       == 0  # Got 0
```

③ 基于LLM虚拟执行的子模块修复

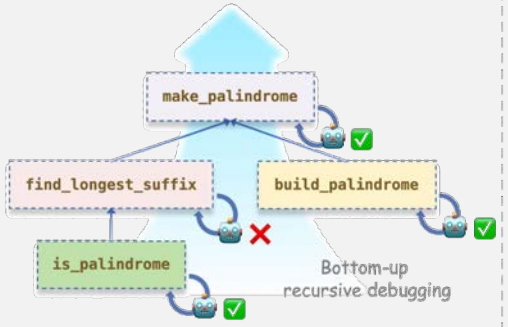
1. `suffix_start = 0`:
   * `suffix_start` is initialized to 0.
2. `for i in range(len('cata'))`:
   * `i` iterates over the the input string length 4.
3. `if is_palindrome('cata'[4:]):`:
   * checks if the substring starting from index 4 is a palindrome.

...

10. `return suffix_start`:
    * finally returns `suffix_start`, which is 3.

We need to stop iterating once a palindrome is found, rather than continuing to the end of the string, by adding a `break`

④ 自底向上修复

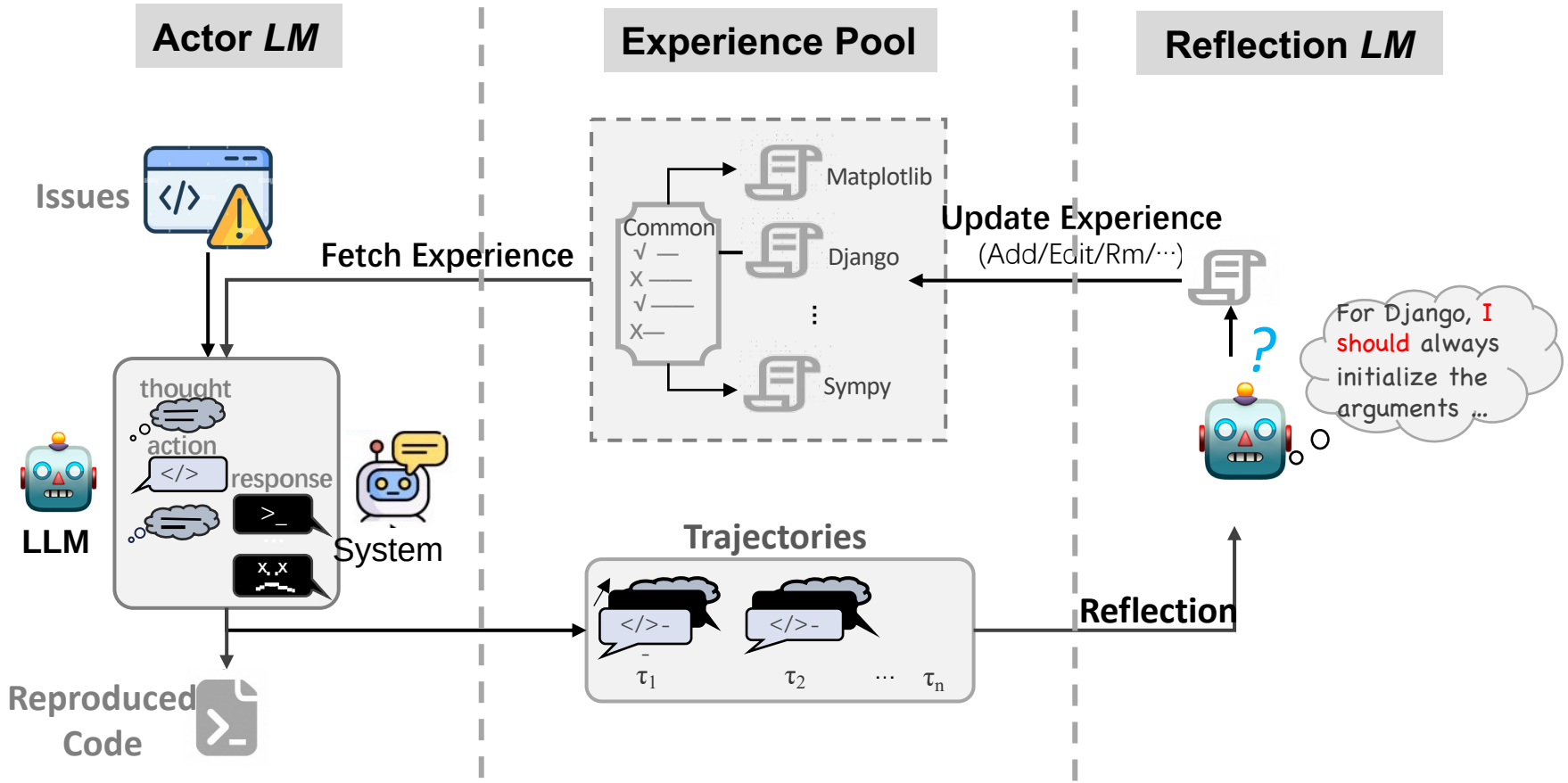


实验结果

Method	HumanEval Acc. (%)	Δ Acc. (%)	HumanEval RSR (%)	MBPP Acc. (%)	Δ Acc. (%)	MBPP RSR (%)
No-Debugging	76.8	--	--	67.2	--	--
Simple Feedback	82.3	+5.5	23.7	69.4	+2.2	6.7
Self-Edit	82.9	+6.1	26.3	71.2	+4.0	12.2
LDB (Block)	84.1	+7.3	31.6	74.0	+6.8	20.7
Self-Debugging (Expl.)	87.2	+10.4	44.7	73.4	+6.2	18.9
Self-Debugging (Trace)	86.0	+9.2	39.5	72.6	+5.3	16.5
Reflexion	90.9	+14.1	60.5	76.6	+9.4	28.7
Our Approach	**94.5**	**+17.7**	**76.3**	**80.0**	**+12.8**	**39.0**

EvoCoder: 基于LLM-Agent反思的Issue代码复现

- 多Agent反思机制 + 层次化经验库

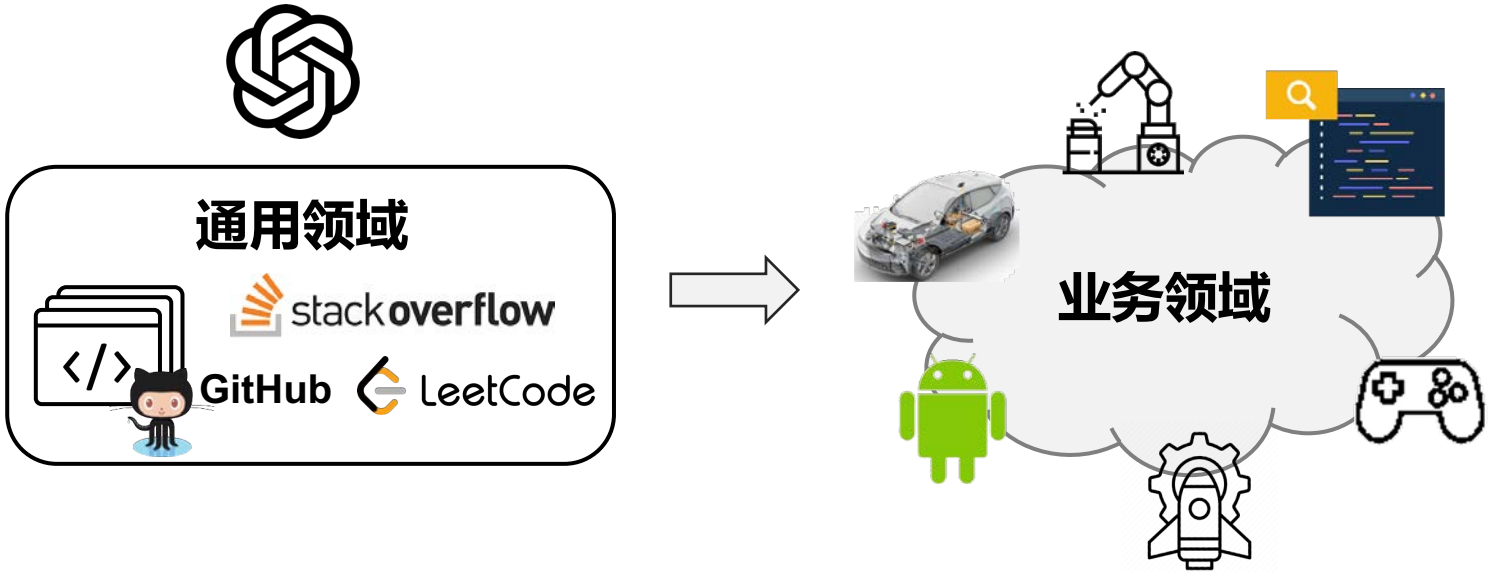


实验结果

- **在问题代码复现上的效果**

Method	Accuracy (%)	
	Model-judged	Human-judged
SWE-agent	26.67	27.00
CodeR	34.00	33.33
LIBRO	45.00	45.00
EvoCoder (Ours)	**54.00 (+9.00)**	**53.33 (+8.33)**
- w/o Action	45.66	46.00
- w/o Repo-Specific Experiences	46.67	46.33
- w/o General Experiences	49.33	49.00

总结



探究
大模型在业务领域代码生成方面的表现

方法
业务知识如何与大模型融合

深入
基于思维链的正则表达式生成

实践
仓库级代码和问题修复

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 · 敦煌站
K+ 思考周®研习社
时间：2025.08.29-30

K+峰会 · 上海站
K+ 金融专场
时间：2025.10.17-18

K+峰会 · 香港站
K+ 思考周®研习社
时间：2025.11.25-26

K+峰会详情

AiDD峰会

AiDD峰会 · 上海站
AI+研发数字峰会
时间：2025.05.17-18

AiDD峰会 · 北京站
AI+研发数字峰会
时间：2025.08.08-09

AiDD峰会 · 深圳站
AI+研发数字峰会
时间：2025.11.28-29

AiDD峰会详情

AiDD AI+研发数字峰会 第5届
AI+ Development Digital summit

利用AI技术深化计算机对现实世界的理解

推动研发进入智能化时代