秀湖会议

逻辑与概率结合的程序合成

汇报人：熊英飞

北京大学计算机学院软件研究所

长聘副教授、研究员

团队正名专用胶片

TreeGen	GrammarT5	DeepSeek
• 首个基于Transformer的代码生成模型 • 千万参数级别效果最好的模型	• 5亿参数以下的效果最好代码生成模型	• 博士生实习创业时开发 • 效果最好的代码生成开源模型，超过ChatGPT 3.5

缺陷修复=错误定位+代码生成
程序合成=（偏向于逻辑正确性的）代码生成

训练代码大模型

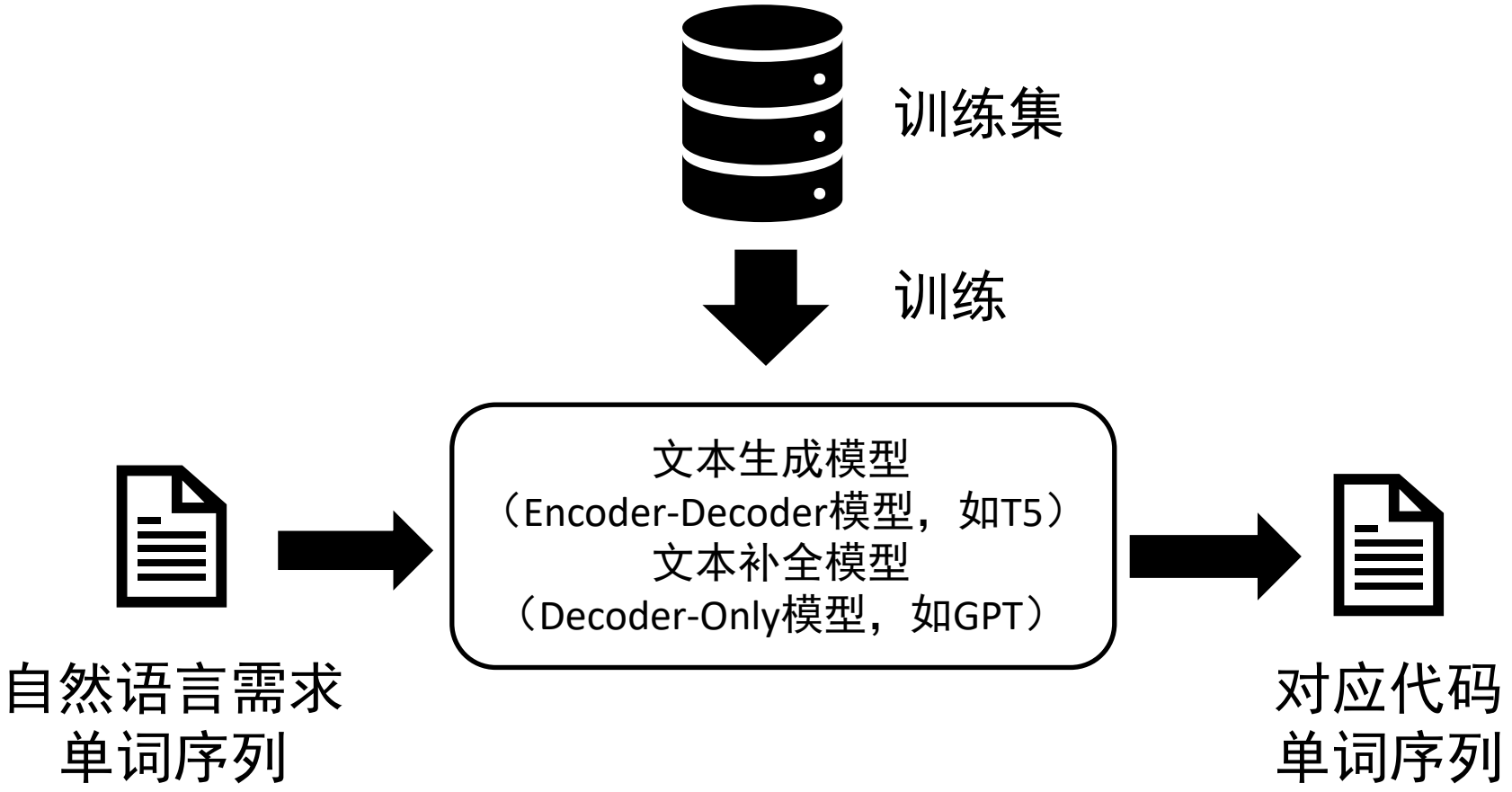
参数越多，效果越好，成本越高

大型模型的训练成本在几千万到数十亿不等

能否保持性能不变降低参数量需求，或者在同样参数的情况下提高模型性能？

LLM把代码当文本处理

通用模型是针对文本训练的，代码当文本处理使得通用模型可以直接应用。



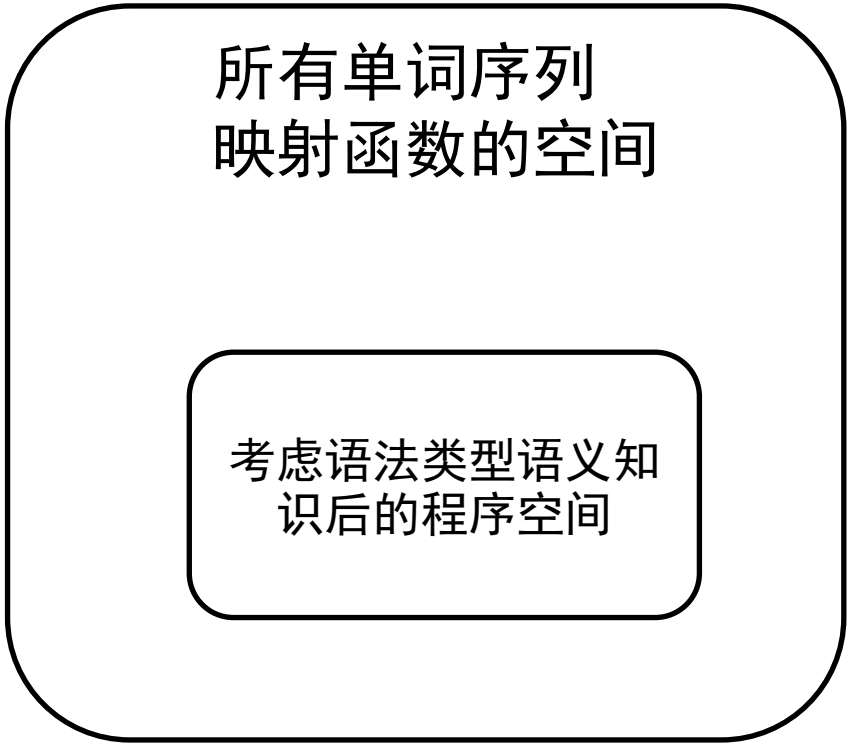
潜在问题：忽略语法类型等领域知识(1/3)

程序有大量语法、类型、语义等领域知识，忽略这些知识使得生成程序的空间变大，学习的难度变高。

语法约束：()+5(不合法

类型约束：1+true不合法

语义约束：期望输出3时返回5



假设空间（计算学习理论）

潜在问题：忽略语法类型等领域知识(2/3)

生成代码时需要用到语法、类型、语义等领域知识。

```
bool and(bool a, bool b) {
    ________________
}
```

赋值语句很常见，多半要填赋值语句

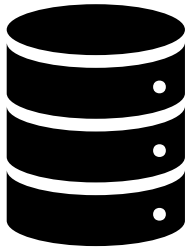
不懂类型的神经网络

参数都是bool形，if语句会更有可能

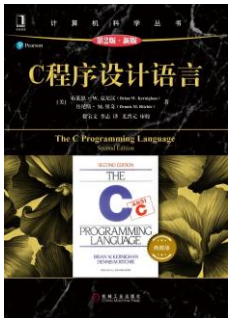
懂类型的神经网络

潜在问题：忽略语法类型等领域知识(3/3)

语法、类型、语义等知识很难直接通过大量代码训练就学到。



海量
代码

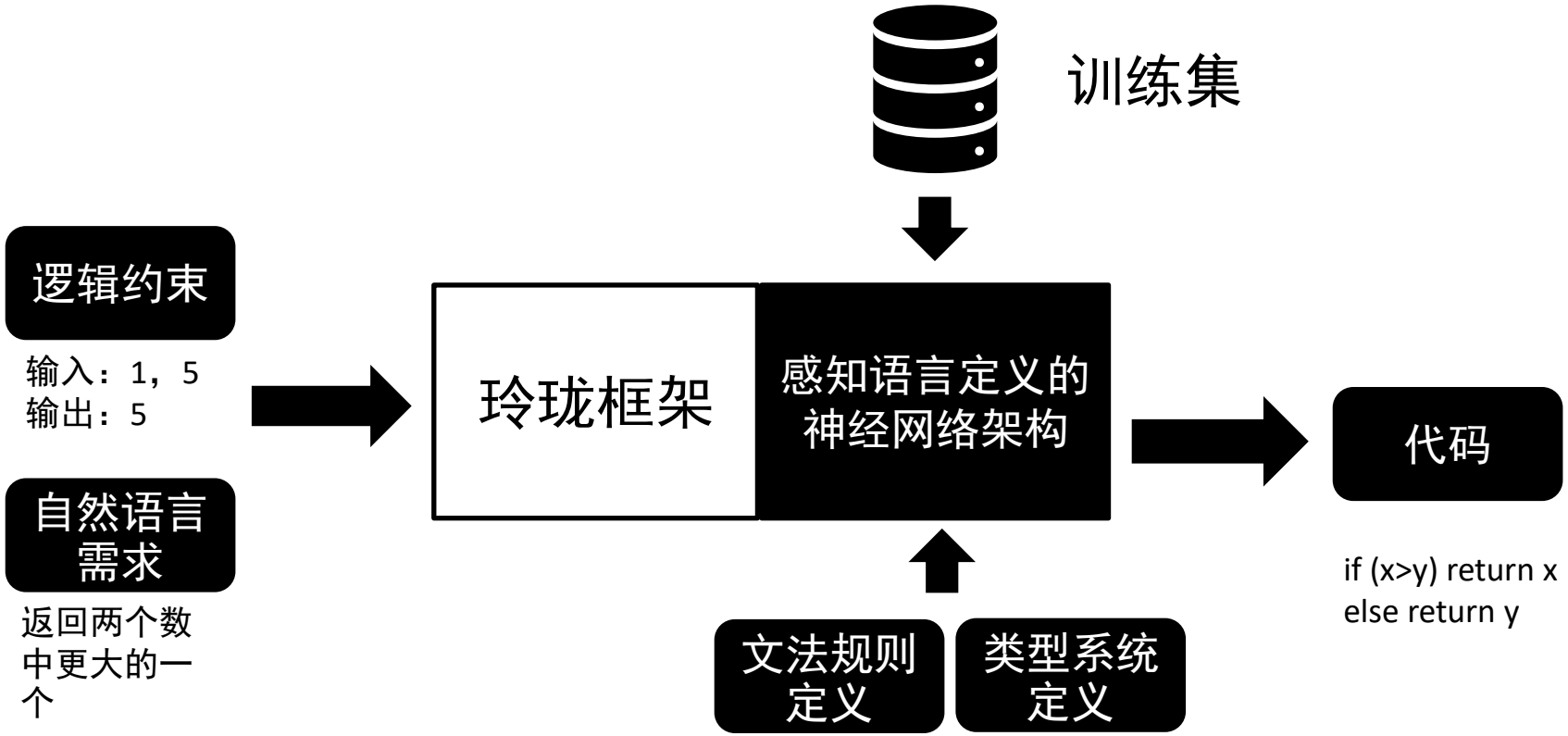


C语言
文法、
类型系统
和语义

逻辑和概率结合的程序合成

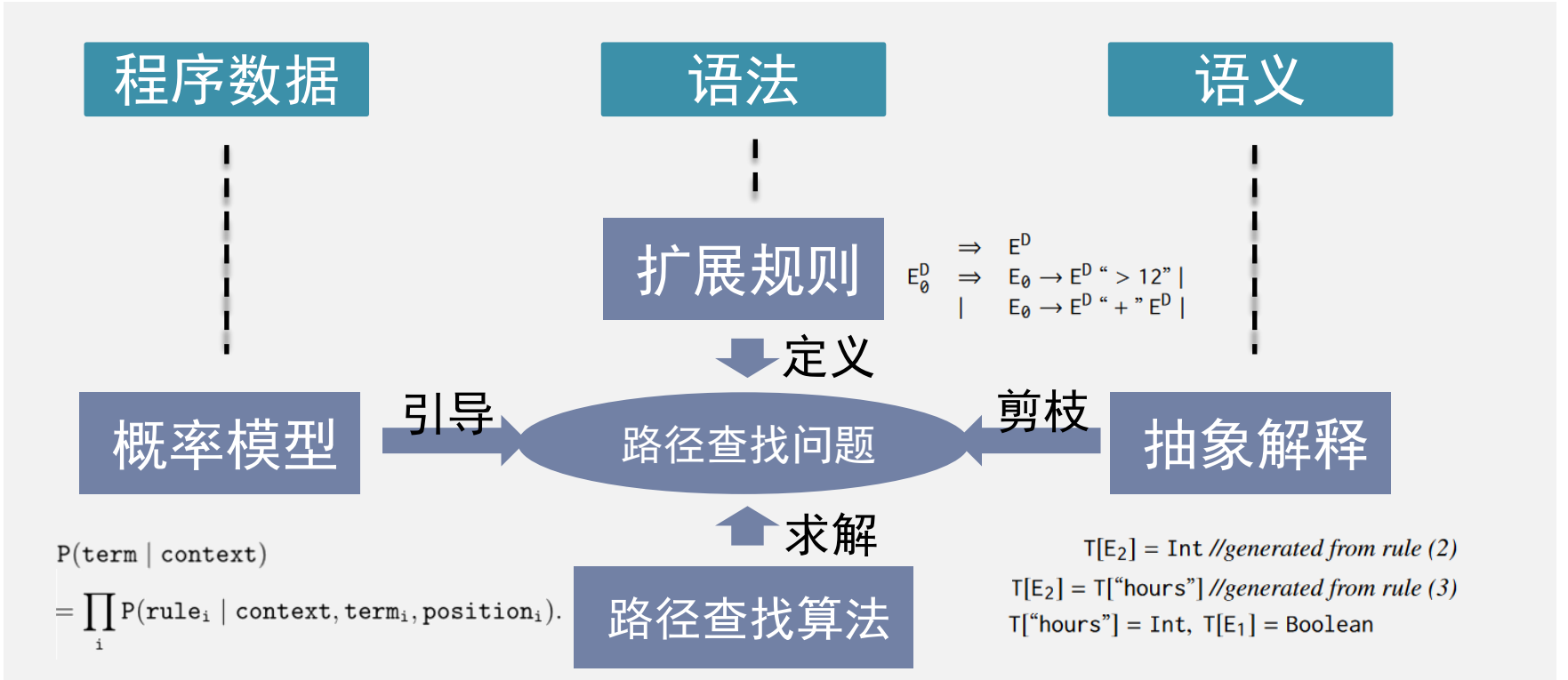
北京大学提出了玲珑框架和感知语言定义的神经网络架构：

- 玲珑框架：保证生成程序(1)满足语言定义(2)概率最高
- 感知语言定义的神经网络架构：引导神经网络学会语言定义



玲珑框架[TOSEM22]

玲珑框架L2S：通过语法规则序列表示程序，剪枝不符合语言定义和语义约束的程序



语法规则序列表示程序

程序　　x+y

单词序列　　x, +, y

规则序列　　r_1, r_2, r_3

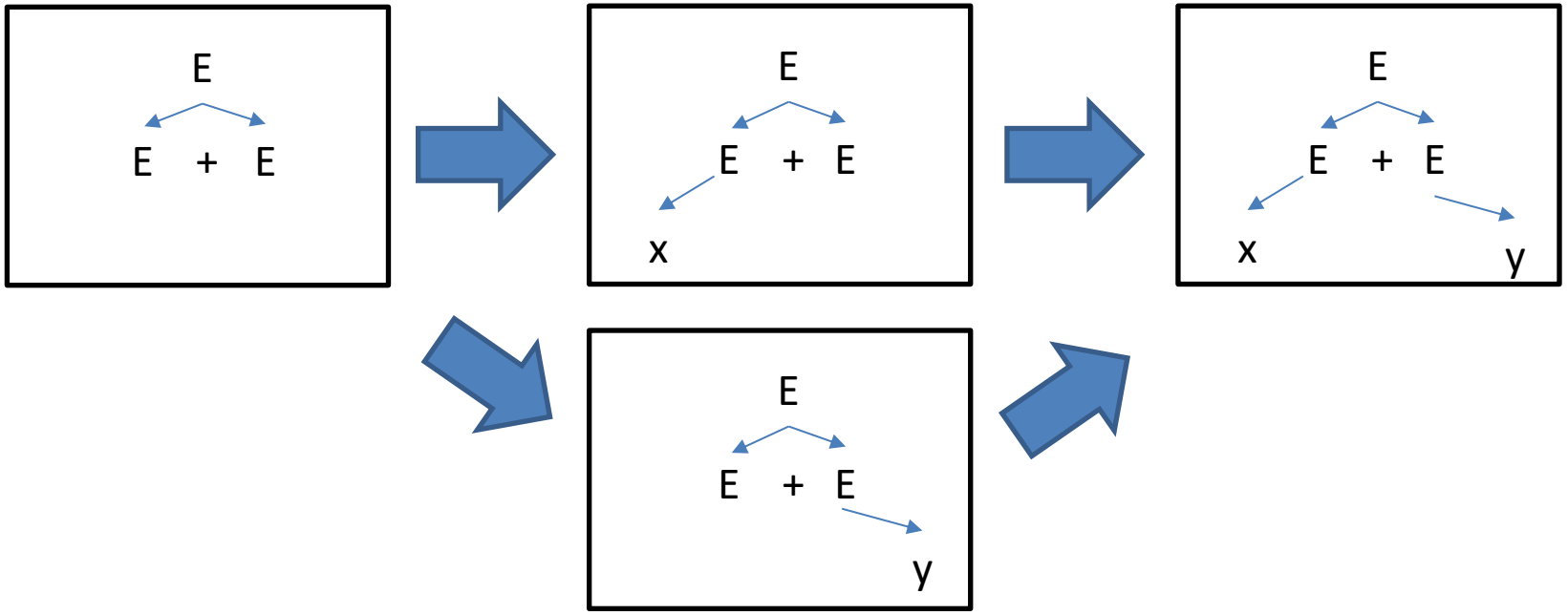
$r_1: E \rightarrow E + E$

$r_2: E \rightarrow x$

$r_3: E \rightarrow y$

保证输出程序符合语法

如何计算程序的概率：问题



同一个程序可能从不同路径到达，是否影响结果？

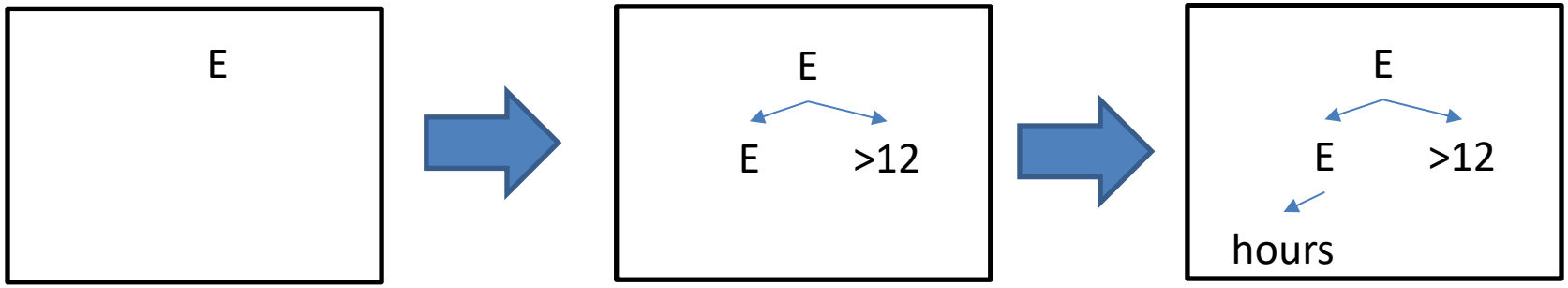
玲珑框架——概率定理

$$P(prog \mid prompt)$$
$$= \prod_i P(\ rule_i \mid prompt, prog_i, position_i\ )$$

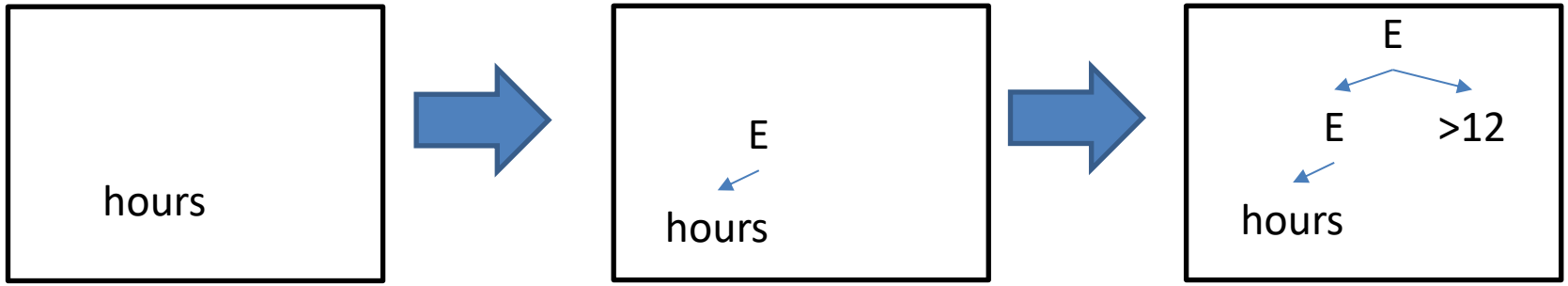
- 程序的概率只和规则选择概率有关，和AST结点展开顺序无关
- 可以根据需要选择合适的顺序生成

超越上下文无关文法的顺序?

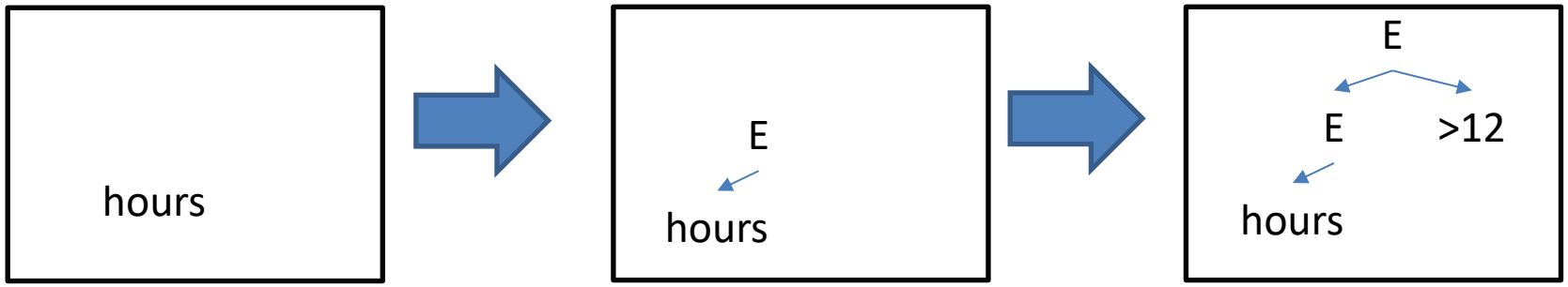
- 自顶向下



- 自底向上



玲珑框架——扩展规则



- 扩展规则：上下文无关文法的扩展，用于支撑任意方向的生成
- 提出了扩展规则树，对应AST
- 提出了扩展规则树和AST的双向转换条件和算法

$$
\begin{aligned}
&\langle E \rightarrow \text{“}hours\text{”}, && \bot\rangle\\
&\langle E \rightarrow \text{“}value\text{”}, && \bot\rangle\\
&\langle E \rightarrow E\ \text{“}>12\text{”}, && 1\rangle\\
&\langle E \rightarrow E\ \text{“}+\text{”}\ E, && 1\rangle\\
&\langle T \rightarrow E, && 1\rangle\\
&\langle E \rightarrow E\ \text{“}>12\text{”}, && 0\rangle\\
&\langle E \rightarrow E\ \text{“}+\text{”}\ E, && 0\rangle\\
&\langle E \rightarrow \text{“}hours\text{”}, && 0\rangle\\
&\langle E \rightarrow \text{“}value\text{”}, && 0\rangle
\end{aligned}
$$

如何保证找到的程序满足语义和类型的要求？

搜索过程中剪枝

- 语义：假设输入变量的取值仅为2，要求输出为3，且文法中只有加号，那么E+E肯定无法满足
- 类型：E+E && E肯定无法满足

基于抽象解释可以对文法规则预分析，可以计算出动态剪枝条件，快速剪枝不能满足规约的部分程序

基于文法的生成可以确保生成的部分程序都有语法结构

采用神经网络实现玲珑框架[AAAI20]

用Transformer实现玲珑框架中的概率模型

- 最早的采用Transformer生成代码的工作
- 将Transformer适配到文法规则上形成TreeGen

	Model	StrAcc	Acc+	BLEU
Plain	LPN (Ling et al. 2016)	6.1	–	67.1
	SEQ2TREE (Dong and Lapata 2016)	1.5	–	53.4
	YN17 (Yin and Neubig 2017)	16.2	~18.2	75.8
	ASN (Rabinovich, Stern, and Klein 2017)	18.2	–	77.6
	ReCode (Hayati et al. 2018)	19.6	–	78.4
	TreeGen-A	**25.8**	**25.8**	**79.3**
Structural	ASN+SUPATT (Rabinovich, Stern, and Klein 2017)	22.7	–	79.2
	SZM19 (Sun et al. 2019)	27.3	30.3	79.6
	TreeGen-B	**31.8**	**33.3**	80.8

TreeGen还被后续研究应用到反编译、代码修复、代码搜索、自动化代码编辑等多个领域，均取得了显著超越SOTA的表现

基于玲珑框架的程序缺陷修复

在玲珑框架下，采用TreeGen来构造概率模型，并用修改操作定义了程序空间

Table 2: Comparison without Perfect Fault Localization

Project	jGenProg	HDRepair	Nopol	CapGen	SketchFix	FixMiner	SimFix	TBar	DLFix	PraPR	AVATAR	Recoder
Chart	0/7	0/2	1/6	4/4	6/8	5/8	4/8	**9/14**	5/12	4/14	5/12	8/14
Closure	0/0	0/7	0/0	0/0	3/5	5/5	6/8	8/12	6/10	12/62	8/12	**17/31**
Lang	0/0	2/6	3/7	5/5	3/4	2/3	**9/13**	5/14	5/12	3/19	5/11	**9/15**
Math	5/18	4/7	1/21	12/16	7/8	12/14	14/26	**18/36**	12/28	6/40	6/13	15/30
Time	0/2	0/1	0/1	0/0	0/1	1/1	1/1	1/3	1/2	0/7	1/3	**2/2**
Mockito	0/0	0/0	0/0	0/0	0/0	0/0	0/0	1/2	1/1	1/6	**2/2**	**2/2**
Total	5/27	6/23	5/35	21/25	19/26	25/31	34/56	42/81	30/65	26/148	27/53	**53/94**
P(%)	18.5	26.1	14.3	**84.0**	73.1	80.6	60.7	51.9	46.2	17.6	50.9	56.4

In the cells, x/y:x denotes the number of correct patches, and y denotes the number of patches that can pass all the test cases.

神经网络修复四年多来首次超过传统修复的效果

录用于ESEC/FSE21，是该会议上引用最多的论文

玲珑框架局限性

在玲珑框架主要从外部加上语法、类型等约束

神经网络并没有全面了解语言中文法规则和类型规则的定义

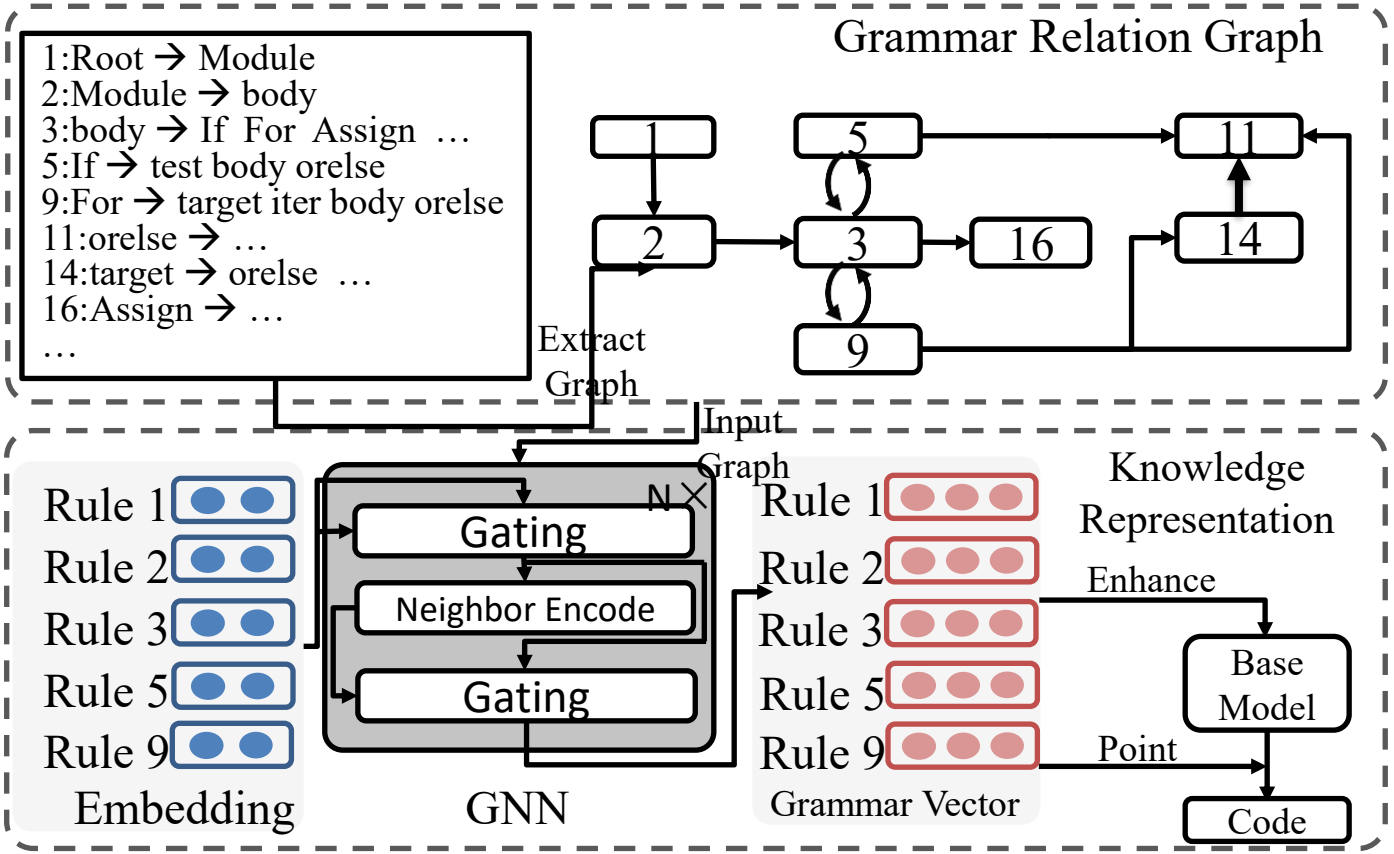
文法规则在神经网络中编码为数字

ifstmt -> ‘if’ ‘(‘ boolExpr ‘)’ stmt	10
whilestat -> ‘while’ ‘(‘ boolExpr ‘)’ stmt	11
boolExpr -> andExpr	12
boolExpr -> orExpr	13

神经网络并不了解这些数字的含义，有可能预测出10, 11这样的序列，但应用10之后是无法应用11的

感知文法定义[IJCAI22]

基于文法规则的前后关联关系构造文法关系图，基于该图构建图神经网络，然后通过该图的训练学出规则的向量表示，类似Word2Vec。



感知文法定义效果

提升TreeGen在不同应用上效果0.8-6.4个百分点。

超越参数量和训练集都更大的预训练模型。

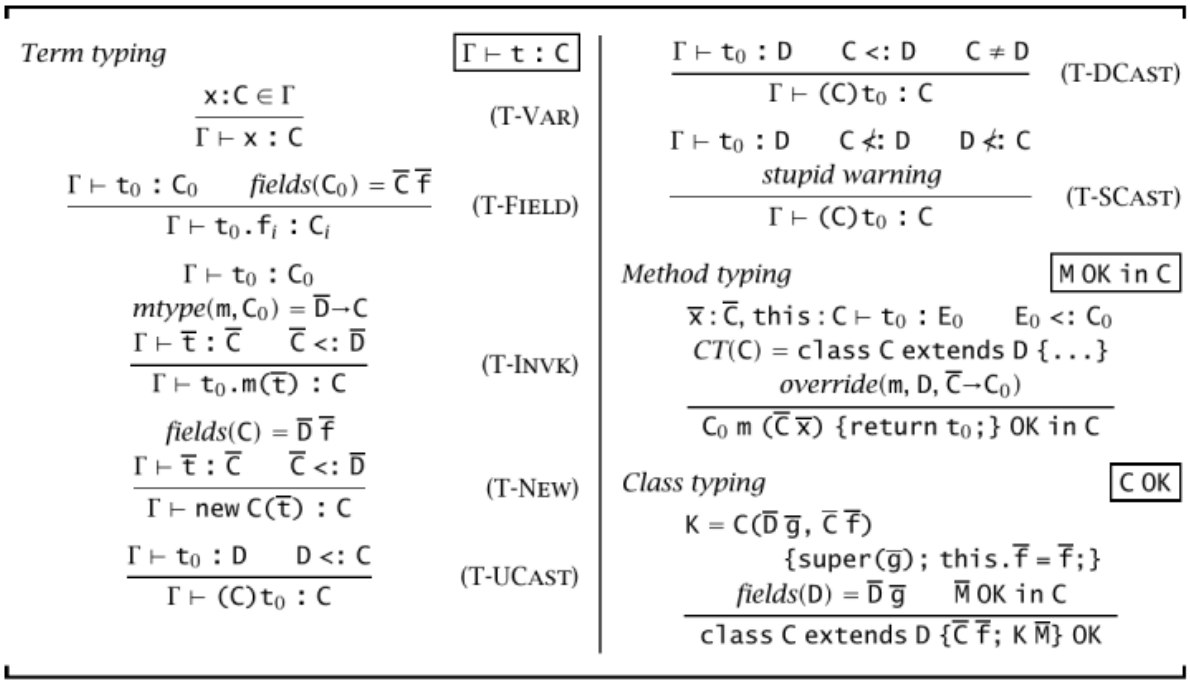
		Code Generation					Semantic Parsing		Regex Synthesis
	Method	**HearthStone**			**Django**	**Concode**	**Atis**	**Job**	**StrReg**
	Metric	**StrAcc**	**BLEU**	**Acc+**	**StrAcc**	**StrAcc**	**ExeAcc**	**ExeAcc**	**DFAAcc**
	KCAZ13 [Kwiatkowski *et al.*, 2013]	-	-	-	-	-	89.0	-	-
	WKZ14 [Wang *et al.*, 2014]	-	-	-	-	-	91.3	90.7	-
Neural Networks	SEQ2TREE [Dong and Lapata, 2016]	-	-	-	-	-	84.6	90.0	-
Neural Networks	ASN+SUPATT [Rabinovich *et al.*, 2017]	22.7	79.2	-	-	-	85.9	**92.9**	-
Neural Networks	TRANX [Yin and Neubig, 2018]	-	-	-	73.7	-	86.3	90.0	-
Neural Networks	Iyer-Simp+200 idoms [Iyer *et al.*, 2018]	-	-	-	-	12.20	-	-	-
Neural Networks	GNN-Edge [Shaw *et al.*, 2019]	-	-	-	-	-	87.1	-	-
Neural Networks	SoftReGex [Park *et al.*, 2019]	-	-	-	-	-	-	-	28.2
Neural Networks	TreeGen [Sun *et al.*, 2020]	30.3±1.061	80.8	33.3	76.4	16.6	89.6±0.329	91.5±0.586	22.5
	GPT-2 [Radford *et al.*, 2019]	16.7	71	18.2	62.3	17.3	84.4	92.1	24.6
	CodeGPT [Lu *et al.*, 2021]	27.3	75.4	30.3	68.9	**18.3**	87.5	92.1	22.49
	TreeGen + Grape	**33.6±1.255**	**85.4**	**36.3**	**77.3**	17.6	**92.16±0.167**	**92.55±0.817**	**28.9**

参数量：TreeGen+Grape: 3510万 GPT-2、CodeGPT：1.1亿

千万参数级别最好模型

感知类型规则［ICSE23］

完整类型系统由多条规则组成，很复杂，难以从数据中直接学会



Term typing $\boxed{\Gamma \vdash t : C}$

$$\frac{x{:}C \in \Gamma}{\Gamma \vdash x : C} \quad \text{(T-Var)}$$

$$\frac{\Gamma \vdash t_0 : C_0 \qquad \mathit{fields}(C_0) = \overline{C}\ \overline{f}}{\Gamma \vdash t_0.f_i : C_i} \quad \text{(T-Field)}$$

$$\frac{\Gamma \vdash t_0 : C_0 \qquad \mathit{mtype}(m, C_0) = \overline{D} \to C \qquad \Gamma \vdash \overline{t} : \overline{C} \qquad \overline{C} <: \overline{D}}{\Gamma \vdash t_0.m(\overline{t}) : C} \quad \text{(T-Invk)}$$

$$\frac{\mathit{fields}(C) = \overline{D}\ \overline{f} \qquad \Gamma \vdash \overline{t} : \overline{C} \qquad \overline{C} <: \overline{D}}{\Gamma \vdash \mathsf{new}\ C(\overline{t}) : C} \quad \text{(T-New)}$$

$$\frac{\Gamma \vdash t_0 : D \qquad D <: C}{\Gamma \vdash (C)t_0 : C} \quad \text{(T-UCast)}$$

$$\frac{\Gamma \vdash t_0 : D \qquad C <: D \qquad C \neq D}{\Gamma \vdash (C)t_0 : C} \quad \text{(T-DCast)}$$

$$\frac{\Gamma \vdash t_0 : D \qquad C \not<: D \qquad D \not<: C \qquad \textit{stupid warning}}{\Gamma \vdash (C)t_0 : C} \quad \text{(T-SCast)}$$

Method typing $\boxed{M\ \mathsf{OK\ in}\ C}$

$$\frac{\overline{x} : \overline{C}, \mathsf{this} : C \vdash t_0 : E_0 \qquad E_0 <: C_0 \qquad \mathit{CT}(C) = \mathsf{class}\ C\ \mathsf{extends}\ D\ \{\ldots\} \qquad \mathit{override}(m, D, \overline{C} \to C_0)}{C_0\ m\ (\overline{C}\ \overline{x})\ \{\mathsf{return}\ t_0;\}\ \mathsf{OK\ in}\ C}$$

Class typing $\boxed{C\ \mathsf{OK}}$

$$\frac{K = C(\overline{D}\ \overline{g}, \overline{C}\ \overline{f})\ \{\mathsf{super}(\overline{g}); \mathsf{this}.\overline{f} = \overline{f};\} \qquad \mathit{fields}(D) = \overline{D}\ \overline{g} \qquad \overline{M}\ \mathsf{OK\ in}\ C}{\mathsf{class}\ C\ \mathsf{extends}\ D\ \{\overline{C}\ \overline{f};\ K\ \overline{M}\}\ \mathsf{OK}}$$

Figure 19-4: Featherweight Java (typing)

修复工具生成的程序中只有30%-40%是类型正确的

感知类型规则-方法

假设单条规则是容易学会的

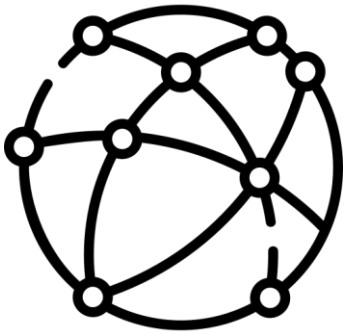
1. 设计数据结构表征学习单条规则所需的信息
2. 修改文法规则记录类型推导的结果

类型规则

$$\frac{\Gamma \vdash v : D \quad \Gamma \vdash t : C \quad C <: D}{\Gamma \vdash v = t : \text{Void}}$$

类型关系

(1)AST和类型
(2)变量和类型
(3)类型之间的子关系



T-Graph

T-Grammar

感知类型规则-效果

应用于缺陷修复，形成Tare方法（参数量3510万），成功修复的数量提升26.5%

Project	Bugs	CapGen	SimFix	TBar	DLFix	Hanabi	Recoder	Recoder-F	Recoder-T	Tare
Chart	26	4/4	4/8	9/14	5/12	3/5	8/14	9/15	8/16	**11/16**
Closure	133	0/0	6/8	8/12	6/10	-/-	13/33	14/36	15/31	**15/29**
Lang	64	5/5	9/13	5/14	5/12	4/4	9/15	9/15	11/23	**13/22**
Math	106	12/16	14/26	18/36	12/28	19/22	15/30	16/31	16/40	**19/42**
Time	26	0/0	1/1	1/3	1/2	2/2	**2/2**	**2/2**	**2/4**	**2/4**
Mockito	38	0/0	0/0	1/2	1/1	-/-	**2/2**	**2/2**	**2/2**	**2/2**
Total	393	21/25	34/56	42/81	30/65	28/33	49/96	52/101	54/116	**62/115**

目前仍是全球效果最好的Java修复工具，刚刚在国际修复比赛上获得Java组冠军
超过其他团队用几亿、几十亿参数预训练模型构建的工具

大型预训练模型[ICSE24]

现代大型代码预训练模型都采用多种编程语言训练。
不同编程语言文法规则、类型规则各不相同。
以上方法能否应用于预训练模型？

<identifier> ::= <initial> | <initial> <more>
<initial> ::= <letter> | _ | $
<more> ::= <final> | <more > <final>
<final> ::= <initial> | <digit>
<letter> ::= a | b | c | … z | A | B| … | Z
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Java文法

Python文法

```
stringliteral   ::=  [stringprefix](shortstring | longstring)
stringprefix    ::=  "r" | "u" | "R" | "U" | "f" | "F"
                     | "fr" | "Fr" | "fR" | "FR" | "rf" | "rF" | "Rf" | "RF"
shortstring     ::=  "'" shortstringitem* "'" | '"' shortstringitem* '"'
longstring      ::=  "'''" longstringitem* "'''" | '"""' longstringitem* '"""'
shortstringitem ::=  shortstringchar | stringescapeseq
longstringitem  ::=  longstringchar | stringescapeseq
shortstringchar ::=  <any source character except "\" or newline or the quote>
longstringchar  ::=  <any source character except "\">
stringescapeseq ::=  "\" <any source character>
```

大型预训练模型

可简单将文法合并为更大的文法

神经网络能学会不同文法规则之间的关联

其他预训练技术（如BPE分词）也能做到和文法兼容

```
Root -> JavaRoot
      | PythonRoot
      | ...
JavaRoot -> Imports Classes
...
```

大型预训练模型

效果全面超越原同规模最好的CodeT5-base模型

同参数量约三倍的CodeT5-Large效果相当

	Natural Language-Based Code Generation							
Models	**Concode**			**Conala**		**Django**		**MBPP**
Metric	**BLEU**	**EM**	**CodeBLEU**	**BLEU**	**EM**	**BLEU**	**EM**	**pass@80**
GPT-C(110M)	30.85	19.85	33.10	30.32	4.80	72.56	68.91	10.40
CodeGPT-adapted(110M)	35.94	20.15	37.27	31.04	4.60	71.24	72.13	12.60
CoTexT(220M)	19.19	19.72	38.13	31.45	6.20	75.91	78.43	14.00
PLBART(220M)	36.69	18.75	38.52	32.44	5.10	72.81	79.12	12.00
CodeT5-small(60M)	38.13	21.55	41.39	31.23	6.00	76.91	81.77	19.20
CodeT5-base(220M)	40.73	22.30	43.2	38.91	8.40	81.40	84.04	24.00
CodeT5-large(770M)	**42.66**	22.65	45.08	39.96	7.40	82.11	83.16	**32.40**
Unixcoder(110M)	38.73	22.65	40.86	36.09	10.20	78.42	75.35	22.40
GrammarT5-small(60M)	38.68	21.25	41.62	39.18	8.00	81.20	82.77	26.00
GrammarT5-base(220M)	42.30	**24.75**	**45.38**	**41.42**	**10.40**	**82.20**	**84.27**	32.00

学生实习成果：DeepSeek

DeepSeek
Coder-33B

朱琪豪

预期2024年6月博士毕业
Grape,Recoder,Tare,DeepSeek Coder第一作者
TreeGen第二作者
字节跳动奖学金（首个软工获得者）
ASE22杰出论文奖、FSE21杰出论文提名
面向高校找工作，欢迎各位老师赐Offer

幻方子公司深度求索开发
全球效果最好开源代码生成模型
其7B版本就超过了ChatGPT3.5

小结

- 人类有几千年探索积累的知识体系
- 信念：仅靠端到端的学习难以还原这样的知识体系
- 程序语法、语义、类型知识可以有效提升模型效果
  - **只高效搜索知识约束下的目标程序空间**
  - **引导神经网络学到这部分知识**
- 未来工作：如何用更通用的方式编码和利用各种不同的知识？