

Impact of Code Language Models on Automated Program Repair

Nan Jiang
Purdue University
West Lafayette, USA
jiang719@purdue.edu

Kevin Liu
Lynbrook High School
San Jose, USA
kevin.bx.liu@gmail.com

Thibaud Lutellier
University of Alberta
Alberta, Canada
lutellie@ualberta.ca

Lin Tan
Purdue University
West Lafayette, USA
lintan@purdue.edu

2023 IEEE/ACM 45rd International Conference on Software Engineering (ICSE)

発表：山村慎太郎

背景 | APR

- | APR：プログラムの不具合を自動で修正する手法
—パッチを自動生成することで、開発者がソフトウェアの信頼性を向上させることを支援
- | 従来手法の中では、DLベースの手法が高い効果を発揮
しかし、
 - ・バグの大部分はまだ修正できない
 - ・十分な数のパッチ候補の生成・検証にはかなり時間がかかるなどの問題点があり、研究段階と実用段階には隔たりがある

背景 | CLM(Code Language Model)

ソースコードの生成や理解に特化した自然言語処理モデル

- 大規模なコードベースを学習データとして、ソースコードの特性を学習(事前学習)
- コード補完やコード生成、コード要約など、コードに関わる様々なタスクに応用される



APRにCLMを用いた研究はほとんど行われていない
→CLMはAPRに良い影響をもたらすか？

目的

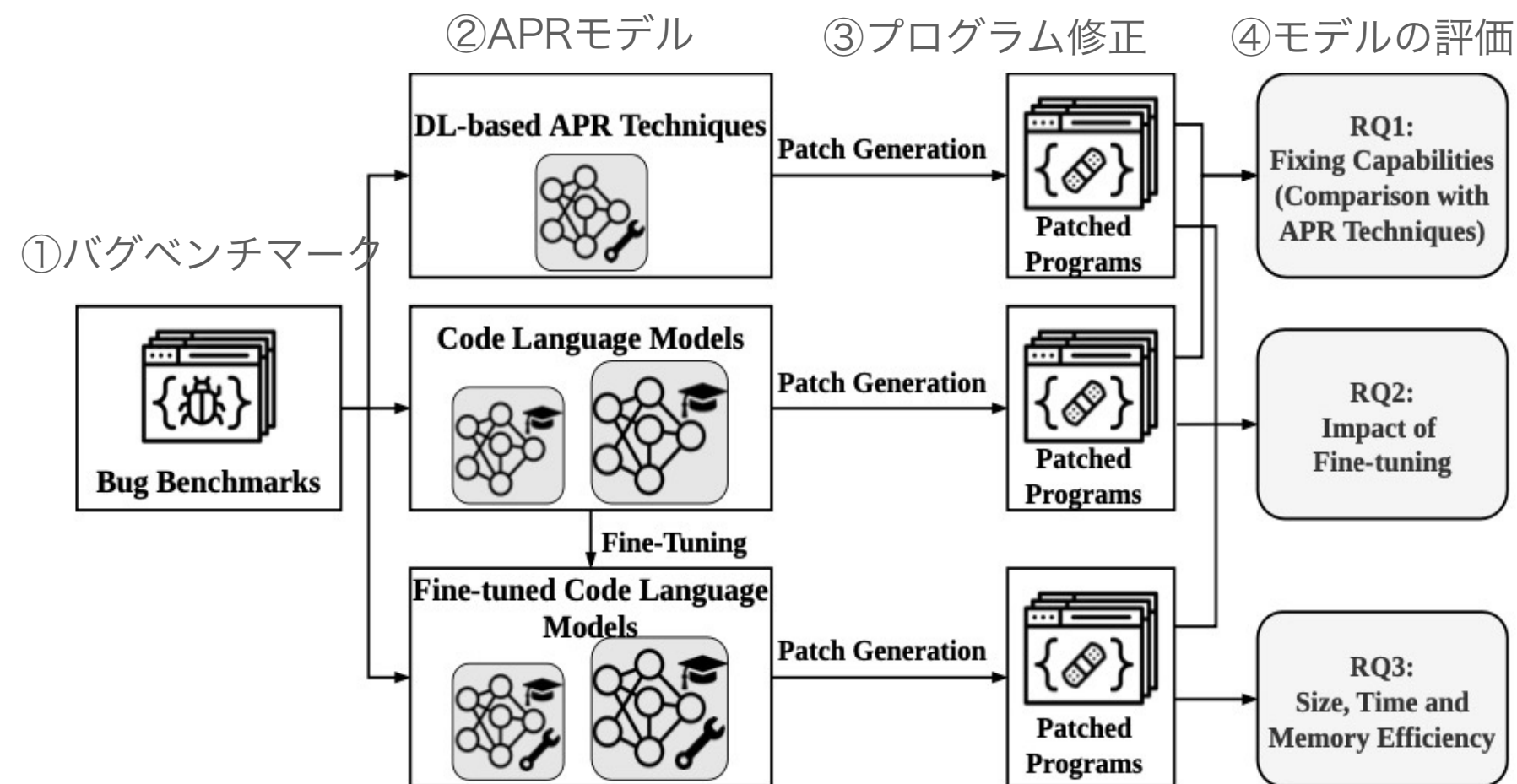
- | CLMはAPRに改善をもたらすか
- | DLモデルとCLMモデルの性能の差は？
- | CLMモデルをAPR用にファインチューニングした際のCLMモデルの性能は？

実験方法

バグベンチマーク

バグのあるプログラムとその修正後のプログラムのセット

APRモデルがそのバグをどれほど修正できるかで性能を評価



概要

- ①ベンチマークを用意
- ②APRモデルを用意
- ③それぞれでパッチ生成
- ④どれほどバグを修正できたかを評価

データセット

| 使用モデル10種類

PLBART : PLBART-base, PLBART-large

CodeT5 : CodeT5-small, CodeT5-base, CodeT5-large

CodeGen : CodeGen-350M, CodeGen-3B, CodeGen-6B

InCoder : InCoder-1B, InCoder6B

| 使用ベンチマーク

Defects4J v1.2 : 有名なJavaプロジェクトから収集

Defects4J v2.0 : ↑の最新バージョン

QuixBugs : 有名なアルゴリズムに関するプログラム

HumanEval-Java : この実験で作成された新たなベンチマーク

→他のベンチマークは事前学習で既習済みの可能性があるため

RQ

| RQ1

最先端のDLベースのモデルとFine-tuning前のCLMのバグ修正能力はどちらの方が優れているか？

| RQ2

CLMモデルは、Fine-tuningによってどれほど性能が向上するか？

| RQ3

それぞれのCLMモデルのサイズ効率，時間効率，メモリ効率の比較

RQ1 | CLMの性能

ベンチマーク種類 バグ数

CLM

DLベースモデル

Benchmarks	#Bugs	PLBART		CodeT5			CodeGen			InCoder		DL-based APR Techniques			
		base	large	small	base	large	350M	2B	6B	1B	6B	CURE	Reward	Recoder	KNOD
Defects4J v1.2	130	13	13	1	0	1	4	11	11	10	16	6	20	24	20
Defects4J v2.0	108	9	8	2	4	1	3	4	8	10	15	6	8	11	13
QuixBugs	40	11	12	3	0	3	7	15	16	14	15	5	7	6	10
HumanEval-Java	164	39	52	3	5	6	30	49	46	40	59	18	22	11	18
Total	442	72	85	9	9	11	44	79	81	74	105	35	57	52	61

バグ修正数(CLM)

バグ修正数(DL)

バグ修正数は、全体的にCLMの方が多い

Fine-tuning以前でも、CLMモデルはDLモデルよりも高性能

CodeT5に限り、修正数は少なかった

APRとは大幅に異なるタスクに対する学習がされているため？

CLMのみ修正できないバグも存在した

Fine-tuning以前は、バグ行をうまく活用できないため？

RQ2 | fine-tuning CLMの性能

Benchmarks	#Bugs	PLBART		small	CodeT5		CodeGen			InCoder	
		base	large		base	large	350M	2B	6B	1B	6B
Defects4J v1.2	130	25 (12)	29 (16)	19 (18)	30 (30)	33 (32)	23 (19)	32 (21)	38 (27)	27 (17)	41 (25)
Defects4J v2.0	108	13 (4)	17 (9)	15 (13)	17 (13)	19 (18)	20 (17)	23 (19)	23 (15)	24 (14)	28 (13)
QuixBugs	40	15 (4)	17 (5)	14 (11)	15 (15)	19 (16)	18 (11)	18 (3)	18 (2)	18 (4)	22 (7)
HumanEval-Java	164	41 (2)	48 (-4)	41 (38)	54 (49)	54 (48)	52 (22)	53 (4)	52 (6)	64 (24)	70 (11)
Total	442	94 (22)	111 (26)	89 (80)	116 (107)	125(114)	96 (69)	126 (47)	131 (50)	133 (59)	161 (56)

()内は, fine-tuning以前のCLMからの増加数

修正できたバグ数は全体的に大幅に向上した
InCoder6Bの修正数はDLモデルを100件以上上回った

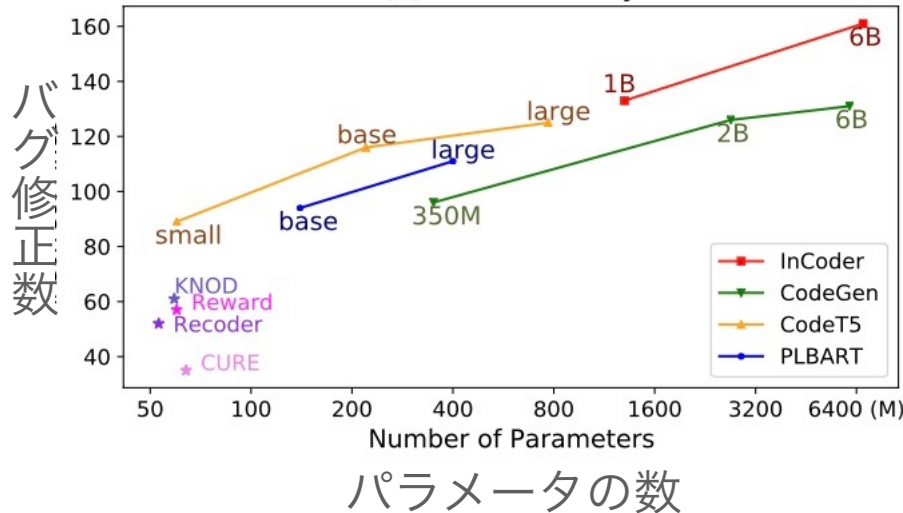
CodeT5もfine-tuningによりDLを上回った

APR用にfine-tuningしたCLMは従来手法を大きく上回ることが示された

RQ3 | サイズ, 時間, メモリ効率

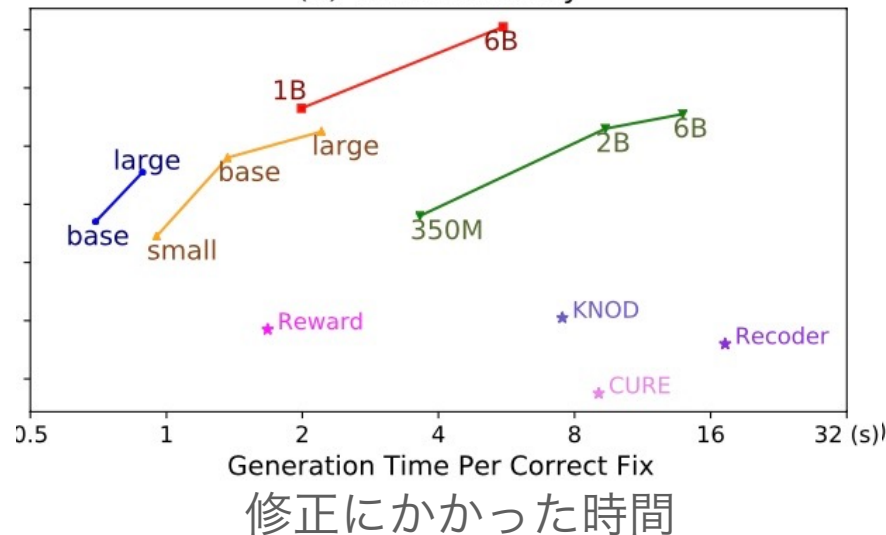
(パラメータ)サイズ効率

(a) Size Efficiency



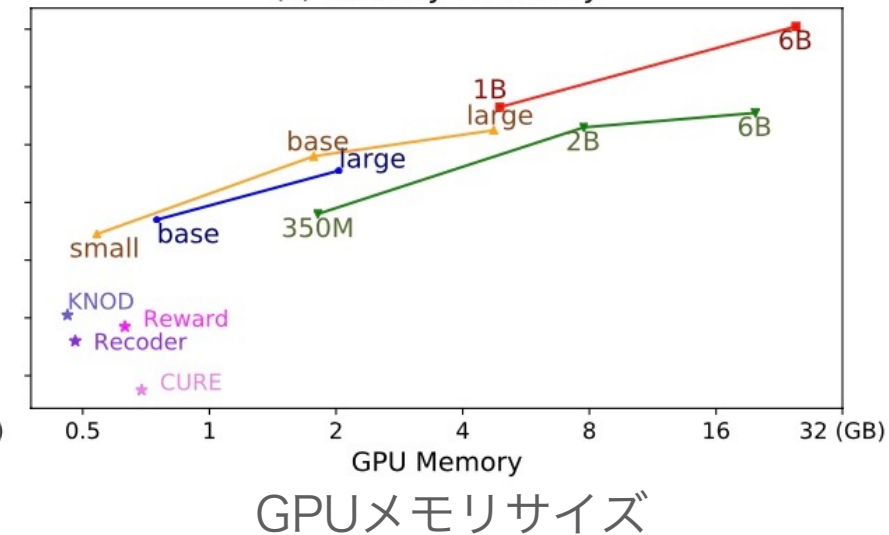
時間効率

(b) Time Efficiency



メモリ効率

(c) Memory Efficiency



| InCoder・CodeT5はサイズ効率に優れる

→より大規模な開発に向いている

| InCoder・CodeT5・PLBARTは時間・メモリ効率に優れる

→限られたリソースの中での開発に向いている

まとめ

| CLMがAPRにもたらす影響を調査

| CLMは事前学習の段階で、最先端のDLベースのAPR性能を上回る

| Fine-tuningにより、更なる性能の向上が見られた

| CLMモデルのサイズ・時間・メモリ効率の比較を行った