

Large Language Models for Test-Free Fault Localization

Aidan Z.H. Yang
aidan@cmu.edu Carnegie Mellon
University Pittsburgh, United
States

Claire Le Goues
clegoues@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, United States

Ruben Martins
rubenm@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, United States

Vincent J. Hellendoorn
vhellendoorn@cmu.edu
Carnegie Mellon University
Pittsburgh, United States

背景 | FL技術

フォルトローカリゼーション

ソフトウェアのバグや不具合の発生個所を特定する
デバッグ作業の一環として行われる

従来はテストがあることが前提になっている

- スペクトラムベース

- ーテストのカバレッジ情報に基づいてコード行の疑わしさのスコアを計算する

- ミューテーションベース

- ーミューテーション解析を用いて特定のコード行がテスト結果に与える影響を評価する

- 機械学習ベース

従来手法の問題点

- | テストカバレッジに依存している

- | データ駆動型の欠陥には使えない

- | ストレートラインブロック内の行を区別できない

- | ステートメントレベルでは限定的な結果

- | 並行性のバグはテストによる発見には適さない

 - 競合状態のようなバグ

目的 | アプローチ

目的

テストケースに依存せず軽量で効果的なフォルトローカリゼーションを実現する

アプローチ

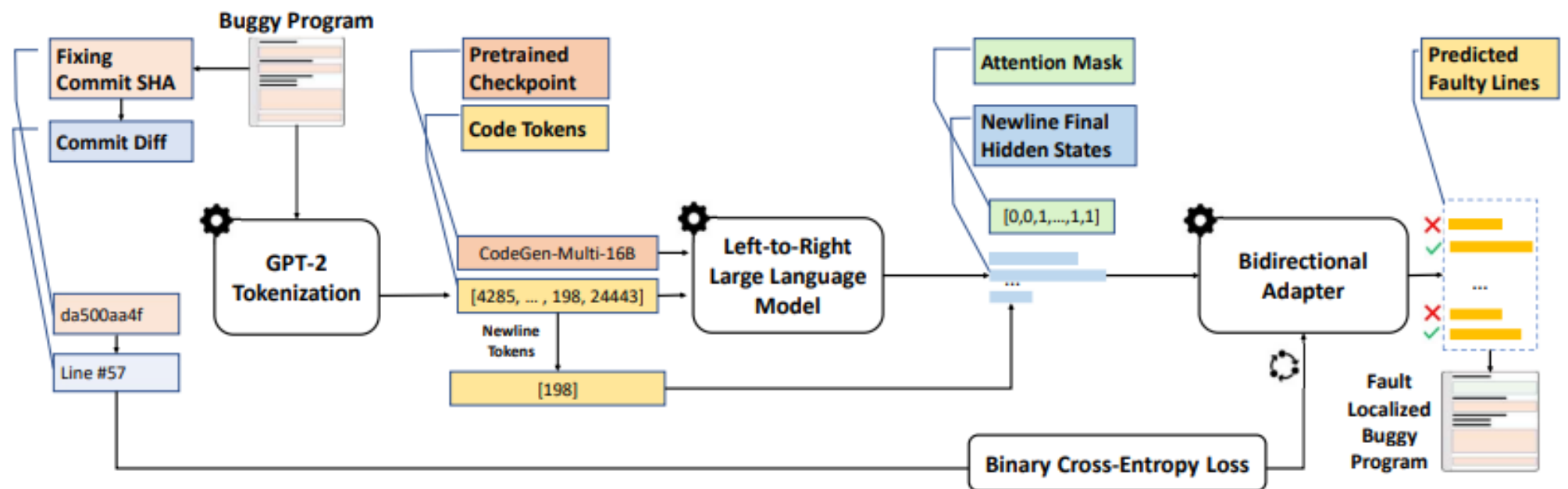
LLMを利用したLLMAOを構築

LLMAO

入力・出力

バグのあるプログラム・疑わしさのスコアの一覧

全体像



トークン化

行ごとにベクトル
表現を取得

バグの可能性を
予測する

行ごとのベクトル表現

Transformerの最終レイヤーから取得できる
このベクトルと対応する位置（〇行目）とペアにして保存

左から右のLLMはバグ予測に適していない
左側のコンテキストの知識のみを反映するため

→双方向にするアダプタが必要

Attentionのマスクをしないことで実現

評価実験

RQ s

1. LLMAOは従来のFL技術と比較してどうか
2. LLMAOの性能は新しいプロジェクトにどれだけ一般化できるか
3. LLMAOの各コンポーネントはその性能にどのように影響するか
4. LLMAOは他の言語やドメインにどれだけ一般化できるか

ベースライン

最先端のステートメントレベルのMLFL

SBFLであるOchiai

評価実験

検証

各データセットで 10 分割交差検証

評価指標

Top-N

最初のN個の中に少なくとも 1 つの欠陥要素が存在する欠陥の数

AUC-ROC

RQ1 | LLMAOは従来のFL技術と比較してどうか

FL type	Technique	Top-1	Top-3	Top-5
SBFL	Ochiai	19 (4.8%)	65 (16.5%)	99 (25.1%)
MLFL	DeepFL	57 (14.4%)	95 (24.1%)	135 (34.2%)
	DeepRL4FL	71 (18.0%)	128 (32.4%)	142 (35.9%)
	TRANSFER-FL	86 (21.8%)	135 (34.2%)	160 (40.5%)
LMFL	LLMAO with CodeGen-350M	82 (20.8%)	106 (26.8%)	126 (31.9%)
	LLMAO with CodeGen-6B	85 (21.5%)	115 (29.1%)	160 (40.5%)
	LLMAO with CodeGen-16B	88 (22.3%)	149 (37.7%)	183 (46.3%)
LMFL, new projects	LLMAO with CodeGen-16B	72 (31.9%)	93 (41.2%)	123(54.4%)
LMFL Ablation	– <i>pretraining</i> (6 layers, trained from scratch)	5 (1.3%)	24 (6.2%)	30 (7.6%)
	– <i>bidirectional adapter</i> (predict directly from CodeGen-16B)	10 (2.6%)	60 (15.2%)	85 (21.5%)

従来手法,アブレーションと比較して良い結果を示した

RQ2 | LLMAOの性能は新しいプロジェクトにどの程度一般化されるか

FL type	Technique	Top-1	Top-3	Top-5
SBFL	Ochiai	19 (4.8%)	65 (16.5%)	99 (25.1%)
MLFL	DeepFL	57 (14.4%)	95 (24.1%)	135 (34.2%)
	DeepRL4FL	71 (18.0%)	128 (32.4%)	142 (35.9%)
	TRANSFER-FL	86 (21.8%)	135 (34.2%)	160 (40.5%)
LMFL	LLMAO with CodeGen-350M	82 (20.8%)	106 (26.8%)	126 (31.9%)
	LLMAO with CodeGen-6B	85 (21.5%)	115 (29.1%)	160 (40.5%)
	LLMAO with CodeGen-16B	88 (22.3%)	149 (37.7%)	183 (46.3%)
LMFL, new projects	LLMAO with CodeGen-16B	72 (31.9%)	93 (41.2%)	123 (54.4%)
LMFL Ablation	–pretraining (6 layers, trained from scratch)	5 (1.3%)	24 (6.2%)	30 (7.6%)
	–bidirectional adapter (predict directly from CodeGen-16B)	10 (2.6%)	60 (15.2%)	85 (21.5%)

事前学習に含まれていないプロジェクトで実験

見知らぬプロジェクトにも遜色ない

RQ4 | LLMAOは他の言語やドメインにどの程度一般化できるか

Metric	<i>BugsInPy</i>	<i>Defects4J</i>	<i>Devign</i>
# lines	76,672	168,960	7,180,160
Top-1	51/493 (10.3%)	88/395 (22.3%)	1478/5260 (28.1%)
Top-3	59/493 (12.0%)	149/395 (37.7%)	2050/5260 (39.0%)
Top-5	75/493 (15.2%)	183/395 (46.3%)	3171/5260 (60.3%)

2つの他の言語と一つの欠陥ドメインで測定

十分なラベル付きデータが与えられると他の言語やドメインにも一般化できる

総括

| LLMベースの手法を提案した

| プログラム解析やテストケースなしで既存の手法を上回った

