

Prompting Is All You Need: Automated Android Bug Replay with Large Language Models

Sidong Feng Chunyang Chen

ICSE 2024

A solid orange horizontal bar at the bottom of the slide.

背景

バグレポート

ユーザーはバグ報告を通じて問題を開発者に知らせる
開発者はそれを基に対応する
保守作業において重要な役割

バグ再現

バグを修正するにはバグを再現する必要がある
手動でバグを再現するプロセスには**時間**と**労力**がかかる

→再現のプロセスを**自動化**したい

バグ再現の自動化

従来の手法

自然言語処理と機械学習を用いた手法

例) ReCDroid ※S2R = Step to Reproduce(再現手順)

バグレポートから単語や句の依存関係を分析し

2 2 種類の文法パターンを定義し、S2Rの要素を抽出する

例えば「名詞句 + “click”」というパターン

問題点

バグレポートの**記述形式**・**単語の選択**・**粒度**に大きく依存

S2Rはユーザーと開発者の間で**認識**・**表現のズレ**がある

様々な表現に対応できない

目的

LLMの力を活用する

自然言語の理解と生成において非常に高い性能
ユーザーの曖昧な記述を理解し、適切に対処できる

本研究の目的

LLMを利用しバグ報告の品質や特徴に依存せず
柔軟にバグを再現する方法を提供する
→開発者の負担を軽減し、バグ修正を迅速に行う

アプローチ

| プロンプトエンジニアリングを用いる

- few-shot learning

少数の入力と出力の例を提示しタスクを理解させる

- chain of thought推論

推論過程を明示的に示すことでより正確な推論を行う

この二つを組み合わせてプロンプトを作る

→ トレーニングが不要

提案手法

AdbGPT

Automatically reproDuce Bugs

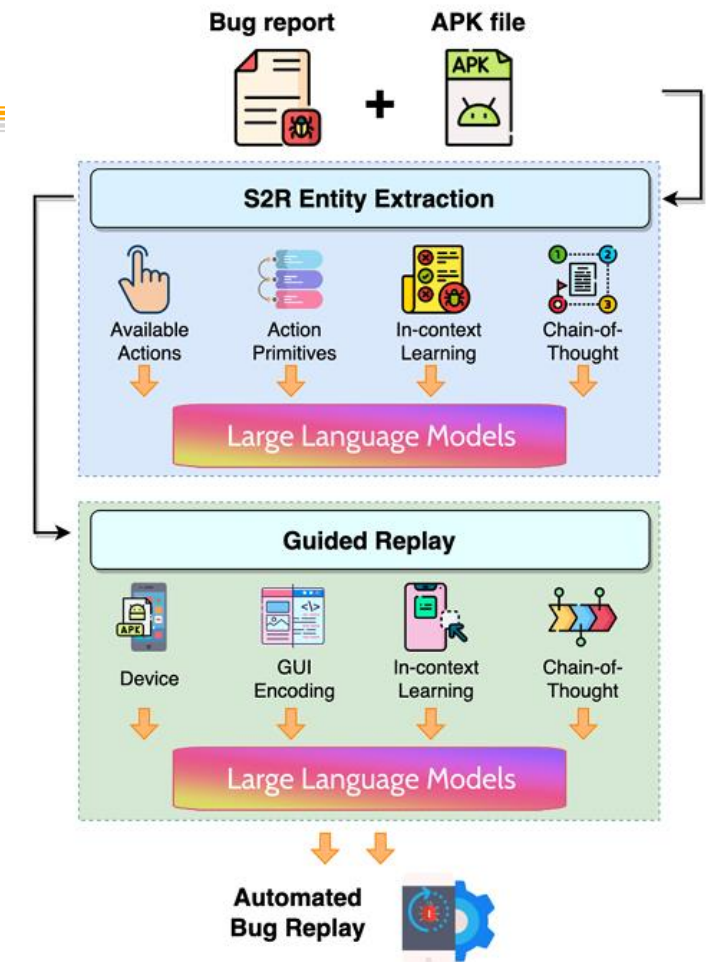


Figure 2: The overview of AdbGPT.

step1, バグ報告からS2Rエンティティを抽出する

S2Rエンティティ：再現手順を表現したもの

step2, 動的なGUIコンポーネントをガイドしながらバグを再現

AdbGPT|Step1

プロンプト

- バグレポートから再現手順を読み取る
- few-shot learningとChain of thoughtの組み合わせ

Table 1: The example of prompt engineering of S2R entity extraction.

PROMPT TYPE	INSTANTIATION
Available actions	<i>Tap, Scroll, Input, Double-tap, Long-tap</i>
Action primitives	<i>[Tap] [Component], [Scroll] [Direction], [Input] [Component] [Value], [Double-tap] [Component], [Long-tap] [Component]</i>
Example input	<i>1. Open bookmark 2. Tap “add new bookmark” and create a name with “a” 3. Create another one with name “b” 4. Click “a” 5. Go back to bookmark after changing name “a” to “b” 6. App crash</i>
Chain-of-Thought	<i>1st step is “Open bookmark”. The action is “open” and the target component is “bookmark”. However, there is no explicit “open” in the Available actions list. Therefore, we select the closest semantic action “tap”. Following the Action primitives, the entity of the first step is [Tap] [“bookmark”]. 2nd step is “Tap ‘add new bookmark’ and create a name with ‘a’”. Due to the conjunction word “and”, this step can be separated into two sub-steps, “Tap ‘add new bookmark’” and “create a name with ‘a’” ... 3rd step is “Create another one with name ‘b’”. Due to its semantic meaning, this step is meant to repeat the previous steps to add another bookmark with name “b”. Therefore, it should actually be the 2nd step ... 4th step is “Click ‘a’” ... 5th step is “Go back to bookmark after changing name ‘a’ to ‘b’”. Due to the conjunction word “after”, this step can be separated into two sub-steps, “Go back to bookmark” and “change name ‘a’ to ‘b’”. The conjunction word “after” also alters the temporal order of the sub-steps, that “change name ‘a’ to ‘b’” should be executed first, followed by “go back to bookmark” ... 6th step is “App crash”. This step does not have any operations.</i>
Example output	<i>Overall, the extracted S2R entities are: 1. [Tap] [“bookmark”] 2. [Tap] [“add new bookmark”] 3. [Input] [“name”] [“a”] 4. [Tap] [“add new bookmark”] 5. [Input] [“name”] [“b”] 6. [Tap] [“a”] 7. [Input] [“name”] [“b”] 8. [Tap] [“back”]</i>

AdbGPT|Step2

GUIエンコーディング

LLMがGUI画面を理解できるようにテキストで表現する
(HTML構文)

プロンプト

例 1 : "Sign in"と"Log in"の表記ゆれ
への対処

例 2 : ステップが抜けている場合

Table 2: The examples of prompt engineering of guide replay.

PROMPT TYPE	INSTANTIATION
Example 1	
GUI encoding	<code><html> ... </html></code>
Example input	<i>If I need to [Tap] ["Sign in"], which component id should I operate on the GUI?</i>
Chain-of-Thought	<i>There is no explicit "Sign in" component in the current GUI screen. However, there is a semantic closest component "Log in" button. The id attribute of "Log in" component is 6. So, we could potentially operate on [id=6] in the screen.</i>
Example output	<code>[id=6]</code>
Example 2	
GUI encoding	<code><html> ... </html></code>
Example input	<i>If I need to [Tap] ["darkmode"], which component id should I operate on the GUI?</i>
Chain-of-Thought	<i>There is no explicit and semantic similar "darkmode" component in the current GUI screen, so it appears a [MISSING] step. However, "darkmode" could be related to the "display" button in the screen. The id attribute of "display" component is 1. So, we could potentially operate on [id=1] component in the screen.</i>
Example output	<code>[MISSING] [id=1]</code>

評価実験

RQs

- 1, S2R エンティティ抽出 (step1)におけるアプローチの精度
- 2, ガイド付きリプレイ (step2)におけるアプローチの精度
- 3, バグ再現におけるアプローチの効率性
- 4, 開発者にとって実世界のバグ再現における
アプローチの有用性

データセット

- 3つの先行研究のデータセットを利用
- 重複したもの, S2Rがテキスト形式でなものを除外し
- 88件のバグレポートをデータセットとして収集

RQ1| S2R エンティティ抽出の性能

ベースライン

ReCDroid MaCa

アブレーション実験（構成要素の一部を取り除く）

ゼロショット学習 思考過程なし

結果

ベースラインを上回った

Table 3: Performance comparison of S2R extraction.

Method	Acc. of Step Extraction	Acc. of Entity Extraction			
		Action	Component	Input	Direction
ReCDroid	49.5%	65.9%	45.4%	30.8%	42.9%
MaCa	51.1%	70.1%	41.6%	33.3%	50.0%
AdbGPT w/o Few	83.9%	81.9%	84.5%	87.1%	85.7%
AdbGPT w/o CoT	86.9%	83.6%	87.6%	87.1%	92.9%
AdbGPT	90.4%	91.1%	90.0%	89.7%	92.9%

RQ1 | ベースラインとの比較

ベースラインだけ誤ったもの

- ・ 形式の違い
- ・ 文脈の考慮
- ・ 異なる表現

1. Settings (General) -> Notifications
2. Tap "Sound", then select "None", press "OK"

- Press "paperclip" button to "Add media". Select first image.
- Repeat to add second image.

Go to setting, click Location Service, choose Network, open Update distance, ...

提案手法で誤ったもの

- ・ 表現が曖昧

 osmeditor4android Issue#637

...
2. Set key "shop" and value to "", max. age to "0"

RQ2 | バグリプレイの精度

結果

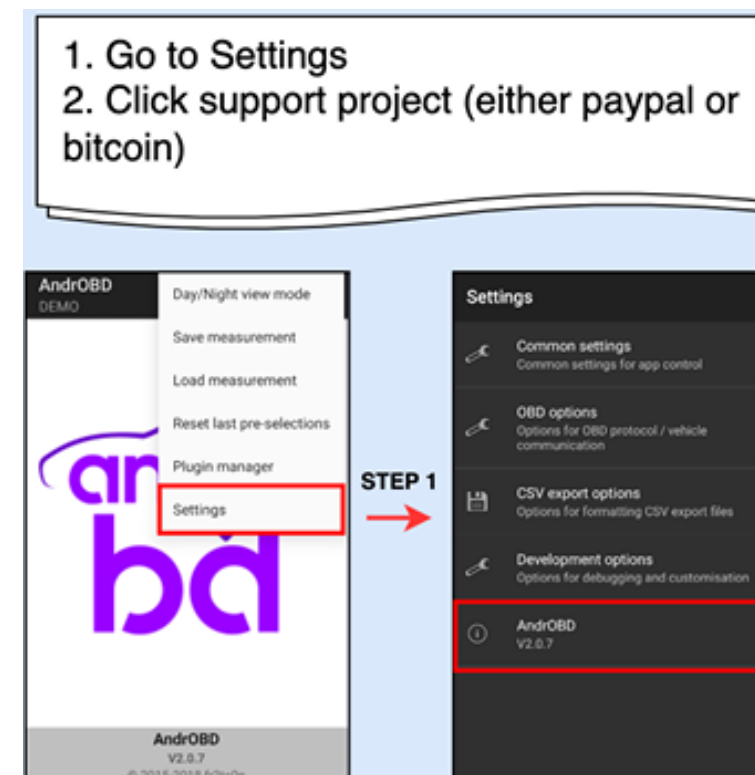
	左から ReCDroid	AdbGPT w/o Few	AdbGPT w/o CoT	AdbGPT	
Reproducibility	-	45.8%	58.3%	62.5%	81.3%

提案手法は他の手法に比べ高い精度を示した

→ Few-shot学習、Chain of thought推論は有効である

ベースラインの失敗原因
ステップの欠落

コンポーネントの不一致



RQ3 | 実行効率

結果

実行時間を調べる

Method	S2R Entity Extraction (sec)	Guided Replay (sec)
ReCDroid	3.75	1357.17
MaCa	2.18	-
AdbGPT	2.11	253.64

既存手法に比べて少ない時間で実行できる
ReCDroidは欠陥ステップが原因で遅くなる

RQ4 | AdbGPTの有用性

実験方法

大学院生5人とプロのアプリ開発者 3 人に調査
被験者がバグレポートからバグを再現するまでの時間を記録
自動バグ再現を見せフィードバックを求める

結果

参加者より早く再現できた

Bug	No. Steps	Participants		AdbGPT	
		Success	Time (sec)	Success	Time (sec)
KISS	2	75%	423.1	100%	262.9
NeuroLab	2	100%	396.5	100%	228.0
Alibaba	5	87.5%	545.7	100%	301.2
Wechat	7	100%	493.3	100%	205.1
Anki	8	100%	544.8	100%	349.7
Average	-	92.5%	480.7	100%	269.4

実務上で使用することを強く支持した（平均4.5/5.0点）

総括・所感

LLMを利用した自動バグ再現手法を提案した

大量のデータを必要とせずプロンプトエンジニアリングによって実現した

自動バグ再現を正確に高速に行うことができた

所感

プロンプトエンジニアリングだけで実現していて面白かった

アンケート調査が8人は少なくないか？

