

exLong: Generating Exceptional Behavior Tests with Large Language Models

Jiyang Zhang and Yu Liu and Pengyu Nie and Junyi Jessy Li
and Milos Gligoric

ICSE 2025

概要

タイトル

- exLong: 大規模言語モデルによる例外的な振る舞いをテストするテストケースの生成

背景

- JavaやPythonなどのメジャーなプログラミング言語には例外処理が用意されている.
- 開発者は例外処理が正しく行われているかどうか確かめるための例外行動テスト (EBTs) を行う.

概要

例外処理

```
1  class SearchCommandParser extends CommandParser {
2      static final char CHR_SPACE = ' ';
3      static final char CHR_CR = '\r';
4
5      public SearchTerm searchTerm(ImapRequestLineReader
6          request) throws ProtocolException {
7          ...
8          char next;
9          while ((next = request.nextChar()) != '\n' &&
10             next != CHR_CR) {
11             next = request.consumeAll(CHR_SPACE);
12             if (isAtomSpecial(next)) {
13                 if (next == '(') {
14                     ...
15                 } else if (next == ')') {
16                     ...
17                     } else {          target throw statement
18                         throw new IllegalStateException("Unsupported atom
19                             special char <" + next + ">");
20                     }
21             }
22             ...
23         }
24     }
25     return handleOperators();
26 }
```

(a) Method under test: searchTerm.

概要

例外行動テスト (EBT)

```
1  public class SearchCommandParserTest {
2      private SearchTerm parse(String line)
3          throws ProtocolException {
4      final byte[] bytes = (line.endsWith("\n")?line:(line
5          + '\n')).getBytes();
6      ByteArrayInputStream ins = new ByteArrayInputStream(
7          bytes);
8      return new SearchCommandParser().searchTerm(new
9          ImapRequestLineReader(ins, null));
10 }
11
12 @Test(expected = IllegalStateException.class)
13 public void testUnsupportedAtomSpecialChar()
14     throws ProtocolException {
15     parse("*");
16 }
```

method under test

exceptional behavior test

(b) Exceptional behavior test written using JUnit 4 that covers the highlighted statement above.

概要

現状と課題

- 開発者はテストを作成する際に、主にハッピーパス（例外処理を含まない動作）のテストに時間をかける.
- 実際のテストにもEBTsは含まれているが割合はかなり少ない.
- 既存のテスト生成システムにおいても、実在するテストに含まれるEBTsの割合が少ない影響でEBTsを生成するのには不向き.

目的

- 例外行動テストという新しいタスクを定義したうえで、それに適したデータセットを作成し、EBTsを自動生成するシステムを開発する.

アプローチ

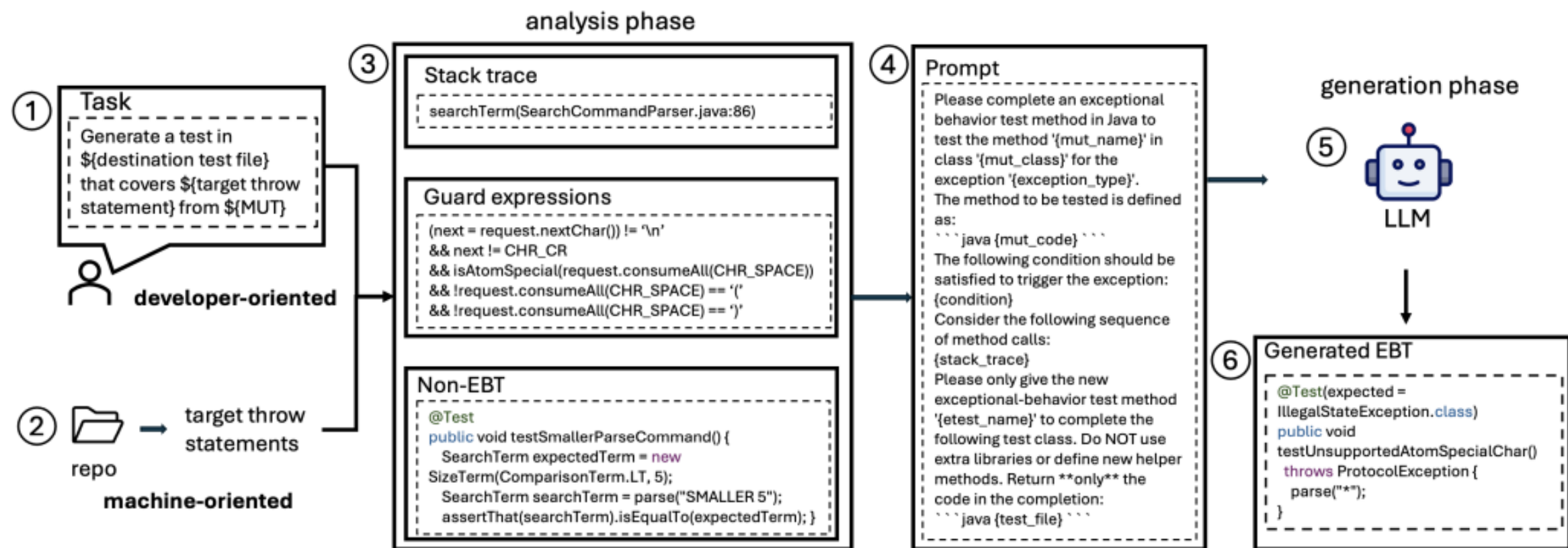


Fig. 2: Overview of EXLONG. Two use cases for EXLONG: (1) developer-oriented use case and (2) machine-oriented use case. In the developer-oriented use case, a developer specifies the method under test, a target throw statement and a destination test file and asks EXLONG to generate an EBT that covers the target throw statement. In the machine-oriented use case, a developer provides an entire repository to EXLONG.

ユースケース

開発者指向ユースケース

- 開発者はテスト対象メソッド, ターゲットthrow文, テスト出力先を指定する.
- exLongは, 対象throw文をトリガーするEBTを生成して出力する.
- 対象メソッドにターゲットthrow文が存在しない場合. メソッド呼び出し可能なメソッドを静的解析し, 登場するthrow文のリストを出力する.

ユースケース

マシン指向ユースケース

- 開発者はリポジトリ全体をシステムに提供する.
- exLongは, すべてのメソッドとそのメソッド内のthrow文を見つける.
- さらにテスト出力先も決定したうえで, メソッド, throw文, 出力先の組を作成する.
- それらのすべての組に対してそれぞれ一つのEBTを生成する.

exLong

| 学習フェーズ

- FTを行うために、以下の情報を正解データから抽出する.

1. スタックトレース

- どうやって例外処理にたどり着くかの記録（足跡）

2. ガード式

- 足跡をたどるときに真でなければならなかった条件式

3. 通常のテスト

- テストコードの書き方やスタイルなどの見本

exLong

| 推論フェーズ

1. 通常のテストの足跡を地図として利用する

- 通常のテストをすべて実行し、そのスタックトレースを記録することで地図を作成する

2. 目的地を決める

3. 目的地へのルートを探す

- 「地図」の中から、目的のthrow文へたどり着くルートを見つける

4. ヒントを組み立てる

- ルートからガード式を導くことで学習データと同じ情報がそろう

5. テストを生成する

評価

データセット

- CodeSearchNetから収集.
- 562プロジェクトから111,230件のテストを収集.
(うちEBTsは12,574件)
- 821タイプの例外を網羅.

TABLE I: Statistics of the collected dataset. #MUTs is the number of unique method under test; #Exception Types is the number of unique exception types tested by the EBTs.

	#Projects	#Tests	#EBTs	#MUTs	#Exception Types
All	562	111,230	12,574	6,250	821
Train	501	100,030	11,182	5,508	725
Valid	29	5,298	550	279	66
Eval	32	5,902	842	-	-

評価

TABLE II: Statistics of the evaluation dataset for developer-oriented use case. #MUT is the number of unique method under test; #Exception Types is the number of unique exception types tested by the EBTs; #Throw Statements is the number of unique throw statements covered by the EBTs.

	#EBTs	#MUT	#Exception Types	#Throw Statements
Developer-Oriented	434	267	41	278

TABLE III: Statistics of the evaluation dataset for machine-oriented use case. #Throw Statements is the number of target throw statements we extracted from the repository according to Section III-B2. #Exception Types is the number of unique exception types thrown by the target throw statements.

	#Throw Statements	#Exception Types
Machine-Oriented	649	81

評価

| 評価指標（開発者指向）

1. 類似性ベース（見た目のそっくり度）

- BLUEスコアやレーベンシュタイン距離などを使用して評価

2. 機能的正確性（本当の正しさ）

- Compilable%（コンパイル可能率）
 - エラーなくコンパイルできるか.
- Runnable%（実行可能率）
 - コンパイル後に、問題なく実行できるか.
- ThrowCov%（スロー文カバレッジ率）
 - 最も厳しい指標で、実行した結果狙い通りの例外をちゃんと発生させられたかを測る.

評価

評価指標（マシン指向）

- ThrowCov%（スロー文カバレッジ率）
 - リポジトリで指定されたすべてのスロー文のうちどれだけ成功したか.

評価

RQ1 開発者が特定のテストを作りたい場合, EXLONGは他のAIモデル (GPT-3.5, CAT-LM) と比べてどれくらい優秀か?

TABLE IV: Results on developer-oriented use case with ground-truth EBT's name in the prompt.

Models	BLEU	CodeBLEU	EditSim	xMatch	Compilable%	Matched-E%	Runnable%	ThrowCov%
GPT3.5-few-shot	56.61	64.28	82.30	14.98	75.12	100.00	61.29	48.39
CAT-LM	53.49	59.79	78.91	9.83	71.83	100.00	36.64	30.03
CodeLlama zero-shot	38.23	48.63	66.96	5.53	57.30	96.02	40.17	33.87
EXLONG	63.13	67.49	85.32	19.05	82.10	100.00	67.36	59.45
EXLONG-sample	70.01	74.08	90.09	15.28	93.54	100.00	81.29	71.28

TABLE V: Results on developer-oriented use case without ground-truth EBT's name in the prompt.

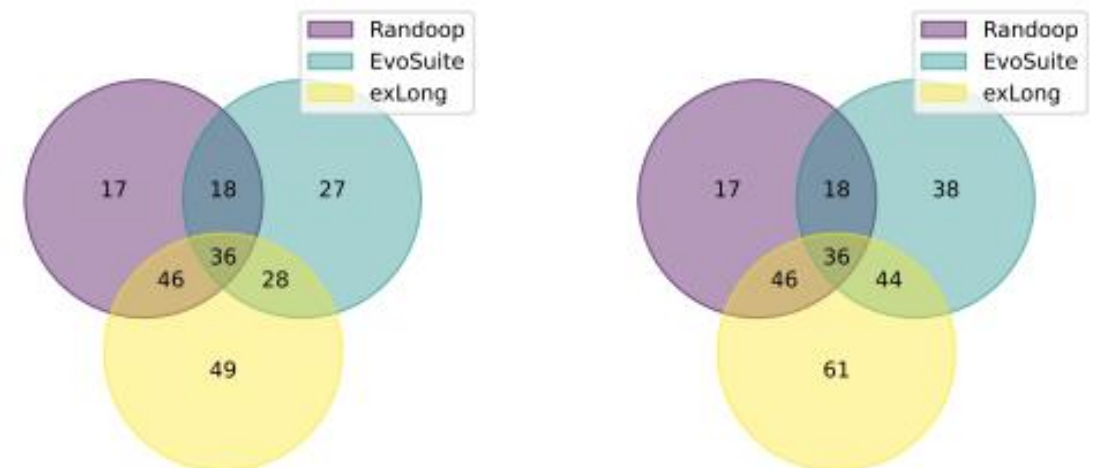
Models	BLEU	CodeBLEU	EditSim	xMatch	Compilable%	Matched-E%	Runnable%	ThrowCov%
GPT3.5-few-shot	38.67	49.33	69.32	0.00	82.83	100.00	57.58	32.32
CAT-LM	37.25	48.47	68.18	1.23	70.83	100.00	23.96	16.44
CodeLlama zero-shot	26.93	37.38	61.88	0.15	57.22	96.91	41.32	33.49
EXLONG	46.66	55.36	79.76	2.07	82.26	100.00	69.89	59.83
EXLONG-sample	50.73	59.38	82.83	1.92	89.40	100.00	78.80	67.67

評価

RQ5 プロジェクト全体の例外を自動でテストする場合，EXLONGは従来のテスト生成ツール（Randoop, EvoSuite）と比べてどれくらい優秀か？

TABLE IX: Throw statements coverage rate for EXLONG, Randoop and EvoSuite.

Tools	ThrowCov%	
	Subset Projects	All Projects
EXLONG	29.72	28.81
EvoSuite	20.37	20.95
Randoop	21.87	18.95



(a) Covered throw statements on a subset of 27 projects.

(b) Covered throw statements on all 30 projects.

Fig. 4: Venn diagram that shows target throw statements coverage by EXLONG, Randoop, and EvoSuite.

まとめと所感

- 例外処理が適切に行われているかを確かめるテストコードを生成する.
- 他の手法と比べると高い精度を発揮したものの、例外処理テストに焦点を当てたシステムがほかにないたため、今後より良いものが開発される可能性は十分ある.
- 外部ライブラリが関わりと失敗するという点は自分の研究と同じである.