6A-01

LLMを利用した自然言語による仕様からテストコードの自動生成

Automatic test code generation from natural language specifications using LLM.

小野塚荘一[*1]

Soichi Onozuka

[*1] 日本アイ・ビー・エム株式会社
IBM Japan #1

Robust programing code generation from non-English natural language, applying prompt technology with large language model (LLM) extensively developed in resent years. However, the generated code faces robustness issues. In this work I propose applying prompt to generate well trained Python code and use Python built-in-function to generate robust test code.

1. はじめに

2023年Large Language Model (LLM)が急速に普及に普及し, GitHub Copilotのようにプログラム開発者も利用その利用が拡大することで，作業の効率化を実現している． [Denny 2023]，しかし工数の大きいテスト工程でテスト自動化コードの開発ではLLMの利用は普及していない．また一般的なテキスト(テキスト)からコード生成の頑健性の調査で自然言語のわずかな変更がセマンティクスにおいて著しく異なるコードを生成する結果になることが指摘されている．[Yan 2023]

本研究では，LLMを活用し日本語のテキストから頑健性の高いソフトウェア・テスト・コード（テストコード）を生成するプロンプト手法を提案する．

研究問題: 頑健性のあるテスト・コードとはどのようなものか.

2. 提案手法

近年のLLMの発展によって様々な種類のコードが生成できるようになったが，頑健性はコード生成よりも抽象構文木(Abstract Syntax Tree: AST)生成の方が高いことがわかっている．[THATTE 2023] このためテキストからASTを生成することに取り組んだ．ASTが生成できればコードの変数とその依存関係を取得することが可能になる．[Guo 2021]

テスト・コードでは入力データ，検証期待値は値を変更して繰り返し実行するため，テストコードから変数や定数を分離することは必須であると考えた．

本研究では上記について，テキストからテストコードの生成はプロンプトで，テストコードからASTさらにデータの分離はPythonの組み込み関数により実現した．

連絡先：小野塚荘一，日本アイ・ビー・エム株式会社，sonozuka@jp.ibm.com

3. 実施結果

http://localhost で Chrome を起動する。
http://localhost start Chrome.

from selenium import webdriver
# Initialize the WebDriver for Chrome 	
driver = webdriver.Chrome(executable_path=""path/to/chromedriver"")
# Navigate to the web page where the button is located 	
driver.get(""http://localhost"")

Module(body=[Assign(targets=[Name(id='driver', ctx=Store())], value=Call(func=Attribute(value=Name(id='webdriver', ctx=Load()), attr='Chrome', value=Constant(value='path/to/chromedriver'))])), Expr(value=Call(args=[Constant(value='http://localhost')],
webdriver.Chrome : path/to/chromedriver driver.get : http://localhost

表1 上から順番に元となった日本語テキスト，プロンプトにより生成された英語テキスト，プロンプトにより生成されたSelenium Pythonテストコード，Pythonの関数で生成したAST, PythonのAST関数でASTから分離したデータ

```
prompt:  [INST] <>
    Web driver, selenium, python code. you are python developer. insert proper indent by tab code and carriage return. Don't forget indenting.
    indenting is very important for python code. Don't generate comment. Must follow pep8 style.
 <> [/INST][English] :
    Press the selection button to switch to the query screen.
    Webdriver Selenium Python code.
    def operation():
        driver=webdriver.Chrome(
        executable_path="path/to/chromedriver")
        driver.get("
```

表 2 英語テキストから Selenium Python を生成したプロンプト

```
ast_obj = ast.parse(python_code)
ast_str = ast.dump(ast_obj, indent=4)
```

```
mport ast
# get constant value from AST
ast_constant_visited = []
def visit_Constant(self, node: ast.Constant):
    ast_constant_visited.append(node.value)
    self.generic_visit(node)
    return node
```

表 3 上から順番に Python コードから AST を取得, AST からデータ（Constant 定数）を取得

モデルは llma2-70b を利用した．テストコードはテスト自動化ツール Selenium のコード生成を行なった．Selenium は様々なプログラミング言語に対応しているが LLM によるコード生成の調査で LLM が最もよく学習していると想定され [FaiWong 2023] Python を選択した．また Python は標準で AST を生成できるクラスを実装していることから，AST の生成はプロンプトに依らず，Python の ast クラスの組み込み関数で生成した．さらに ast の組み込み関数を利用してデータ（Constant 定数）を取得することに成功した．

4. 考察

プロンプトと Python の組み込み関数により，高い頑健性で AST とデータを分離して生成することができた. 本研究で Python をテストコード, AST 生成, データ分離に利用したことが成功の要因と考えられる．Python は表 4 のように学習件数が Java と比較すると一桁多く，また組み込みの ast は visitor design pattern で実装[Martin 2004]されており, 容易かつ頑健にデータの取得ができた.

Code Search	Training examples	Dev queries	Testing queries	Candidate codes
Go	167,288	7,325	8,122	28,120
Java	164,923	5,183	10,955	40,347
JavaScript	58,025	3,885	3,291	13,981
PHP	241,241	12,982	14,014	52,660
Python	251,820	13,914	14,918	43,827
Ruby	24,927	1,400	1,261	4,360

Table 7: Data statistics about the filtered dataset. For each query in the development and testing sets, the answer is retrieved from the whole candidate codes (i.e. the last row).

表 4: GRAPHCODEBERT: PRE-TRAINING CODE REPRESENTATIONS WITH DATA FLOW [Guo 2021]より引用

5. 結論

本研究では，テスト自動化ツールで実行可能なコードの自動生成についてプロンプトと Python の組み込み関数を利用することで頑健なコードとデータの生成方法を示した．

今後の課題として，より大規模なテストコードの生成について頑健性，解釈可能性，説明可能性を検討することが挙げられる．

参考文献

[Denny 2023] Paul Denny,Viraj Kumar, Nasser Giacaman: Prompt Engineering for Solving CS1 Problems Using Natural Language, SIGCSE 2023, March 15–18, 2023.

[Yan 2023] Ming Yan, Junjie Chen, Jie M. Zhang, Xuejie Cao, Chen Yang, Mark Harman: COCO: Testing Code Generation Systems via Concretized, Conference'17, July 2017.

[THATTE 2023] SHREE THATTE: TRANSFORMER-BASED PROGRAM SYNTHESIS THROUGH ABSTRACT SYNTAX TREES, THE DEGREE OF MASTER OF SCIENCE UNIVERSITY OF MASSACHUSETT LOWELL FEBRUARY 2023.

[Guo 2021] Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, Michele Tufano, Shao Kun Deng, Colin Clement, Dawn Drain, Neel Sundaresan, Jian Yin, Daxin Jiang, Ming Zhou: GRAPHCODEBERT: PRE-TRAINING CODE REPRESENTATIONS WITH DATA FLOW, ICLR 2021

[FaiWong 2023] Man-FaiWong , Shangxin Guo, Ching-Nam Hang , Siu-Wai Ho, Chee-Wei Tan: Natural Language Generation and Understanding of Big Code for AI-Assisted Programming: A Review, Entropy 2023, 25, 888

[Fowler 2010] Martin Fowler: Domain Specific Languages, Addison-Wesley Professional

[Selviandro 2023] Khaerunnisa, Nungki Selviandro, Rosa Reska Riskiana: Comparative Study of Robot Framework and Cucumber as BDD Automated Testing Tools, Ultimatics : Jurnal Teknik Informatika, Vol. 15, No. 1, June 2023

[Martin 2004] Robert C. Martin: The Principles, Patterns, and Practices of Agile Software Development, April 12, 2004, Prentice Hall