LLMs for Software Testing and Maintenance

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Software Testing

- Software testing is the process of evaluating if a software product works as expected (dynamic analysis)
- Common classification
  - Unit Testing
  - Integration Testing
  - Functional Testing
  - Acceptance Testing

Software Maintenance

- Software maintenance is the process of modifying, updating, and optimizing software after delivery

- Common classification
  - *Corrective Maintenance* (e.g., fix a bug)
  - *Perfective Maintenance* (e.g., add a feature)
  - *Adaptive Maintenance*
    (e.g., adapt to a new platform)

SLR on LLMs for SE

Large Language Models for Software Engineering: A Systematic Literature Review

XINYI HOU and YANJIE ZHAO, Huazhong University of Science and Technology, Wuhan, China
YUE LIU, Monash University, Melbourne, Australia
ZHOU YANG, Singapore Management University, Singapore
KAILONG WANG, Huazhong University of Science and Technology, Wuhan, China
LI LI, Beihang University, Beijing, China
XIAPU LUO, The Hong Kong Polytechnic University, Hong Kong, China
DAVID LO, Singapore Management University, Singapore
JOHN GRUNDY, Monash University, Melbourne, Australia
HAOYU WANG, Huazhong University of Science and Technology, Wuhan, China

Large Language Models (LLMs) have significantly impacted numerous domains, including Software Engineering (SE). Many recent publications have explored LLMs applied to various SE tasks. Nevertheless, a comprehensive understanding of the application, effects, and possible limitations of LLMs on SE is still in its early stages. To bridge this gap, we conducted a Systematic Literature Review (SLR) on LLM4SE, with a particular focus on understanding how LLMs can be exploited to optimize processes and outcomes. We selected and analyzed 395 research articles from January 2017 to January 2024 to answer four key Research Questions (RQs). In RQ1, we categorize different LLMs that have been employed in SE tasks, characterizing their distinctive features and uses. In RQ2, we analyze the methods used in data collection, pre-processing,

LLMs for Software Engineering

- Number of studies for each SE activity
  - Software development (247)
  - **Software maintenance (99)**
  - **Software quality assurance (66)**
  - Requirements engineering (17)
  - Software design (4)
  - Software management (3)

X. Hou *et al*. "**Large Language Models for Software Engineering: A Systematic Literature Review**". TOSEM, 2024.

Software Testing and Maintenance

SE activity	SE task		Total
Software quality assurance	Vulnerability detection (18)	Test generation (17)	66
	Bug localization (5)	Verification (5)	
	Testing automation (4)	Fault localization (3)	
	Defect detection (2)	GUI testing (2)	
	Static analysis (2)	Binary taint analysis (1)	
	Compiler fuzzing (1)	Decompilation (1)	
	Invariant prediction (1)	Malicious code localization (1)	
	Mobile app crash detection (1)	Resource leak detection (1)	
	Test prediction (1)		
Software maintenance	Program repair (35)	Code clone detection (8)	99
	Code review (7)	Debugging (4)	
	Bug reproduction (3)	Review/commit/code classification (3)	
	Duplicate bug report detection (3)	Logging (3)	
	Log parsing (3)	Code revision (2)	
	Sentiment analysis (3)	Vulnerability repair (2)	
	API misuses repair (1)	Bug prediction (1)	
	Bug triage (1)	Code coverage prediction (1)	
	Code review explained (1)	Code-Review defects repair (1)	
	Crash bug repair (1)	Dockerfile Repair (1)	
	Incivility detection (1)	Patch correctness prediction (1)	
	Patch detection (1)	Program merge conflicts repair (1)	
	Rename Refactoring (1)	Tag recommendation (1)	

Vulnerability Detection

- Traditional static vulnerability detection methods are limited by predefined rules
- Tang *et al*. propose CSGVD
  - Combining Sequence and Graph embedding for Vulnerability Detection
  - For embedding, it relies on CodeBERT - a pre-trained programming language model
  - **Results**: CSGVD achieved 60.39% F1-score

W. Tang et al. "**CSGVD: A Deep Learning Approach Combining Sequence and Graph Embedding for Source Code Vulnerability Detection**". JSS 2023.

Unit Test Generation

- A comparison of test suites generated by ChatGPT 3.5 and EvoSuite (SBST)
  - Comparison is based on correctness, readability, code coverage, and found bugs
- Results of the study (Tang *et al*., 2024)
  - EvoSuite reaches higher code coverage than ChatGPT (74.5% vs 55.4%)
  - EvoSuite finds more bugs than ChatGPT

Y. Tang et al. "**ChatGPT vs SBST: A Comparative Assessment of Unit Test Suite Generation**". IEEE TSE 2024.

Automated Program Repair

- Automated Program Repair (APR) aims to help developers automatically fix bugs

- Xia et al. evaluate LLMs for APR
  - They used 5 datasets across 3 languages
  - They compare 9 LLMs vs. 20 APR tools
  - **Results**. LLMs outperform existing APR tools in all datasets

C. Xia et al. "**Automated Program Repair in the Era of Large Pre-Trained Language Models**". ICSE 2023.

Code Clone Detection

- Semantic code clones are two codes implementing the same functionality
- Kitsios et al. perform a comparative study with six models
  - Three task-specific models and three LLMs (Llama 3.3, GPT 4o, and DeepSeek R1)
  - **Results**. F1 of up to 71% for task-specific models and up to 69% for LLMs

K. Kitsios et al. "**Detecting Semantic Clones of Unseen Functionality**". ASE 2025.

Bibliography

- X. Hou *et al*. **Large Language Models for Software Engineering: A Systematic Literature Review**. TOSEM, 2024.
  - 6.5 How Are LLMs Used in Software Quality Assurance?
  - 6.6 How Are LLMs Used in Software Maintenance?