SOFTWARE ENGINEERING EDUCATION IN THE ERA OF LARGE LANGUAGE MODELS: PROMISES AND PERILS

Massimiliano Di Penta
University of Sannio, Italy
dipenta@unisannio.it, @mdipenta
ABOUT ME

• Full professor, University of Sannio

• Research on software engineering

• Chaired the main software engineering conferences: ICSE 2023, ESEC/FSE 2021, ASE 2017, and others

• Associate Editor-in-Chief: IEEE Transactions on Software Engineering, Editor-in-Chief: Journal of Software Evolution and Process, Associate Editor: Empirical Software Engineering Journal
MY MAIN RESEARCH TOPICS

- Software analytics
- Software evolution
- Recommender systems for software engineers
- DevOps
MY TEACHING

- Advanced software engineering (Master)
- Video game development (Master)
- Natural language processing (Bachelor)
HOW MY SOFTWARE ENGINEERING TEACHING USED TO BE

Master course, so students know already about topics such as

• Analysis and design

• (Some) software configuration management and continuous integration
WHAT’S LEFT?
MY (OLD) COURSE GOALS

• Provide advanced software engineering contents mostly focused on software quality
• Stimulate workgroup
• Stimulate investigation on innovative topics
IN THE MEANTIME... HOW IS SOFTWARE ENGINEERING CHANGING?
ICSE 2023 submitted/accepted papers by topic (top 20)

- Machine learning for SE
- Software testing
- Program analysis
- Mining software repositories
- Evolution and maintenance
- SE for machine learning
- Privacy and security
- Human aspects of software engineering
- Reliability and safety
- Debugging and fault localization
- Program repair
- Program comprehension
- Human-computer interaction
- Apps and app store analysis
- Distributed and collaborative software engineering
- Software reuse
- Formal methods
- Programming languages
- Software metrics and prediction models
- Performance analysis and testing

[Graph showing submitted vs. accepted papers by topic]
ICSE 2023 accepted papers at a glance
THE RISE OF DEEP LEARNING APPLICATIONS TO SE…
BETTER HARDWARE...
MORE (CURATED) DATA

- Torrents, e.g., GitTorrent, SOTorrent, TravisTorrent
- Initiatives like World of Code
- Increase of sharing (large) datasets with papers
- Publication reward for dataset and reproducibility
PRETRAINED MODELS

• SE researchers can focus on
  • Relatively small fine tuning from problem specific data
• No need to bother with large, expensive model training
• ICSE 2023 had a full session on this
An Empirical Study on the Usage of Transformer Models for Code Completion

Matteo Cinelli, Nathan Cooper, Luca Pascarella, Antonio Mastropaolo, Emad Aghajani, Denys Poshyvanyk, Massimiliano Di Penta, and Gabriele Bavota

Abstract—Code completion aims at speeding up code writing by predicting the next code token(s) the developer is likely to write. Works in this field focused on improving the accuracy of the generated predictions, with substantial leaps forward made possible by deep learning (DL) models. However, code completion techniques are mostly evaluated in the scenario of predicting the next token to type, with few exceptions pushing the boundaries to the prediction of an entire code statement. Thus, little is known about the performance of state-of-the-art code completion approaches in more challenging scenarios in which, for example, an entire code block must be generated. We present a large-scale study exploring the capabilities of state-of-the-art Transformer-based models in supporting code completion at different granularity levels, including single tokens, one or multiple entire statements, up to entire code blocks (e.g., the iterated block of a for loop).
Large Language Models for Software Engineering: A Systematic Literature Review

XINYI HOU\(^1\), Huazhong University of Science and Technology, China
YANJIE ZHAO\(^2\), Monash University, Australia
YUE LIU, Monash University, Australia
ZHOU YANG, Singapore Management University, Singapore
KAILONG WANG, Huazhong University of Science and Technology, China
LI LI, Beihang University, China
XIAPU LUO\(^3\), The Hong Kong Polytechnic University, China
DAVID LO, Singapore Management University, Singapore
JOHN GRUNDY, Monash University, Australia
HAOYU WANG\(^4\), Huazhong University of Science and Technology, 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 on LLM4SE, with a particular focus on understanding how LLMs can be exploited to optimize processes and outcomes. We collect and analyze 229 research papers from 2017 to 2023 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, preprocessing, and application highlighting the role of well-curated datasets for successful LLM for SE implementation. RQ3 investigates the strategies employed to optimize and evaluate the performance of LLMs in SE. Finally, RQ4 examines the specific SE tasks where LLMs have shown success to date, illustrating their practical contributions to the field. From the answers to these RQs, we discuss the current state-of-the-art and trends, identifying gaps in existing research, and flagging promising areas for future study.

CCS Concepts:
- General and reference → Surveys and overviews.
- Software and its engineering → Software development techniques.
- Computing methodologies → Artificial intelligence.

Additional Key Words and Phrases: Software Engineering, Large Language Model, Survey.
EVERYTHING SEEMS TO GO SO WELL...

- Better solutions to problems handled already
- Ability to solve problems not considered in the past
Can you write a Python program that gather commits from a GitHub repository?

Yes, here's an example Python program that uses the GitHub API to gather the commit history from a repository:

```python
import requests

github_username = 'your_github_username'
repo_name = 'your_repo_name'

# Set up the API endpoint and headers
api_url = f'https://api.github.com/repos/{github_username}/{repo_name}/commits'
headers = {'Accept': 'application/vnd.github.v3+json'}

# Make the API request
response = requests.get(api_url, headers=headers)

# Check if the request was successful
if response.status_code == 200:
    # Process the commit information
```
This function contains a bug. Can you please fix it?

def quicksort(lst, left=0, right=None):
    if right is None:
        right = len(lst) - 1
    l = left
    r = right
    if l <= r:

Sure, here’s a fixed version of the quicksort function:

```python
def quicksort(lst, left=0, right=None):
    if right is None:
        right = len(lst) - 1
    l = left
    r = right
    if l < r:  # change condition to l < r
```

Changes:

- Change the condition in line 6 to `l < r` instead of `l <= r` to avoid an unnecessary recursive call when `l == r`. 
The code you provided appears to be a simple function that copies the contents of a source batch object (`pSrc`) to a destination batch object (`pDest`) using the `memcpy` function.

As written, this code alone does not seem to be inherently vulnerable. However, whether this code is vulnerable or not depends on how it is used in the larger context of the software system. Here are some things to consider:

1. If the size of `batch_obj_t` changes in the future, this function may need to be updated accordingly. If this function is not updated, it could lead to buffer overflows or other memory-related vulnerabilities.
2. The function assumes that the source and destination batch objects have the same size and structure. If this assumption is not true, it could lead to memory corruption vulnerabilities.
3. If the function is called with invalid pointers (e.g., NULL pointers), it could result in undefined behavior or crashes.
CODE REVIEW...

Can you please write a code review for this file:

```python
#! /usr/bin/python3

import json
import sys
import re
import codecs
import datetime

class CommitData:
    def __init__(self, commitId, commitDate):
        self.commitId = commitId
        self.commitDate = commitDate
```

Sure, here’s a code review for the given file:

- The code follows PEP8 style guide for Python code, which is great.
- The shebang `#!/ /usr/bin/python3` at the top of the file indicates that this script should be run with Python3, which is good to see.
- The `CommitData` class is defined, but it’s not used in the code. It can be removed if it’s not going to be used.
...AND EVEN MORE...

```python
def isBug(msg):
    p = re.compile(r'\b(fix|fixed|fixing|bug)\b', re.IGNORECASE)
    if p.search(msg):
        return True
    return False
```

- The `isBug` function is used to check whether a commit message is a bug fix or not. It uses a regular expression to match the words "fix", "fixed", "fixing", or "bug" in the message, which is a reasonable way to check for bug fixes. However, it could potentially match false positives if the message contains these words but not in the context of a bug fix. For example, a commit message that says "Fixed typo in documentation" would be considered a bug fix even though it's not really fixing a bug. So, it's not a perfect method for identifying bug fixes, but it's a good start.
EACH OF US HAS SURELY TRIED IT FOR MANY OTHER TASKS...

- Code completion
- Test generation
- Program translation and refactoring
- Software design and modeling
- ....
LLMS ARE INFLUENCING RESEARCH…

Novel approaches contribute by leveraging:

• Prompt tuning and chaining

• Hybrid models (e.g., Retrieval Augmented Generation)
No More Fine-Tuning? An Experimental Evaluation of Prompt Tuning in Code Intelligence

Chaozheng Wang  
Harbin Institute of Technology  
Shenzhen, China  
wangchaozheng@stu.hit.edu.cn

Yuanhang Yang  
Harbin Institute of Technology  
Shenzhen, China  
yangyuanhang022@gmail.com

Cuiyun Gao∗  
Harbin Institute of Technology  
Shenzhen, China  
gaocuiyun@hit.edu.cn

Yun Peng  
The Chinese University of Hong Kong  
Hong Kong, China  
ypeng@cse.cuhk.edu.hk

Hongyang Zhang  
The University of Newcastle  
Newcastle, Australia  
hongyang.zhang@newcastle.edu.au

Michael R. Lyu  
The Chinese University of Hong Kong  
Hong Kong, China  
lyu@cse.cuhk.edu.hk

ABSTRACT
Pre-trained models have been shown effective in many code intelligence tasks. These models are pre-trained on large-scale unlabeled corpus and then fine-tuned in downstream tasks. However, as the inputs to pre-training and downstream tasks are in different forms, it is hard to fully explore the knowledge of pre-trained models. Besides, the performance of fine-tuning strongly relies on the amount of downstream data, while in practice, the scenarios with scarce data are common. Recent studies in the natural language processing (NLP) field show that prompt tuning, a new paradigm for tuning, alleviates the above issues and achieves promising results in various NLP tasks. In prompt tuning, the prompts inserted during tuning provide task-specific knowledge, which is especially beneficial for tasks with relatively scarce data. In this paper, we empirically evaluate the usage and effect of prompt tuning in code intelligence tasks. We conduct prompt tuning on popular pre-trained models CodeBERT and CodeT5 and experiment with three code intelligence tasks including defect prediction, code summarization, and code translation. Our experimental results show that prompt tuning overcomes the above limitations and achieves promising results.

KEYWORDS
code intelligence, prompt tuning, empirical study

ACM Reference Format:

1 INTRODUCTION
Code intelligence leverages machine learning, especially deep learning (DL) techniques to mine knowledge from large-scale code corpus and build intelligent models for improving the productivity of computer programming. The state-of-the-art DL-based approaches to code intelligence exploit the pre-training and finetuning paradigm [1, 9, 14, 21, 56], in which language models are first pre-trained on a large unlabeled text corpora and then finetuned on downstream datasets.

• Pretrained models may originate from different types of knowledge, e.g. natural language

• Prompting reintroduce such elements during fine-tuning
RETRIEVAL AUGMENTED GENERATION (RAG)

Retrieval Module

Embedding Extraction

knowledge base

prompt + query

prompt + query

retrieved context

prompt + query + context

generated response

LLM
IMPACT ON SE EDUCATION
Impact on SE Education

Favor a responsible, informed integration of LLMs in the software development process.
FROM INTERACTING WITH STACK OVERFLOW...
TOWARD INTERACTING WITH LLMS...
INSTRUCT STUDENTS ABOUT POTENTIALS

• Applications beyond obtaining code snippets
  • Code review, refactoring, program repair, test generation, documentation generation
• Tutorials on prompt engineering and prompt chaining
• Lack of trust
• Intellectual property problems
• Hallucination
LACK OF TRUST

• Without having a source for the recommended code we can’t trust it

• This is therefore very different from conventional reuse

• We need to define suitable code review procedure for AI-generated software

• Different from conventional code review as there’s no human developer to interact with
INTELLECTUAL PROPERTY ISSUES

Lawsuit says OpenAI violated US authors' copyrights to train AI chatbot

By Blake Brittain
June 29, 2023 8:55 PM GMT+2 - Updated 4 months ago

CONSEQUENCES WITH SOFTWARE DEVELOPMENT

• Again no traceability to recommended code

• Even if the LLM were trained with open source code with permissive licenses, lack of attribution remains a serious violation
IF I ASK CHATGPT SOME CODE...

Can you please write python code to get a git log from a GitHub repository?

Sure, you can use the PyGithub library to interact with GitHub repositories and retrieve the Git log. First, you'll need to install the PyGithub library if you haven't already:

```bash
pip install PyGithub
```

Then, you can use the following Python code to get the Git log from a GitHub repository:

```python
from github import Github

github_username = 'YOUR_USERNAME'
```
...IT HAS NO CLUE ABOUT THE LICENSE!

What is the license for this code?

The code I provided is a simple example of how to interact with GitHub using the PyGithub library, and it doesn't include a specific license. Therefore, it doesn't have a separate license and falls under the default copyright laws, which means you can use it for personal or educational purposes.

If you plan to use the code in a commercial or open-source project and want to specify the terms of use, you should consider adding an appropriate license to your project. Common open-source licenses include the MIT License, Apache License, GNU General Public License (GPL), and more. You can choose a license that aligns with your project's goals and requirements.

Always ensure that you comply with the terms of use for any libraries or tools you utilize in your projects.
WHAT ABOUT SOFTWARE BILLS OF MATERIALS?
By the authority vested in me as President by the Constitution and the laws of the United States of America, it is hereby ordered as follows:

Section 1. Policy. The United States faces persistent and increasingly sophisticated malicious cyber campaigns that threaten the public sector, the private sector, and ultimately the American people’s security and privacy. The Federal Government must improve its efforts to identify, deter, protect against, detect, and respond to these actions and actors. The Federal Government must also carefully examine what occurred during any major cyber incident and apply lessons learned. But cybersecurity requires more than government action. Protecting our Nation from malicious cyber actors requires the Federal Government to partner with the private sector. The private sector must adapt to the continuously changing threat environment, ensure its products are built and operate securely, and partner with the Federal Government to foster a more secure cyberspace. In the end, the trust in our digital infrastructure should be governed by law.

the United States Government brought SBOMs to the forefront of digital policy.
ABOUT CODE REVIEW

• Chicken and egg problem
  • LLM can review your code, but YOU should review LLM code

• Again, lack of trust

• No interaction with a human, maybe with the LLM?

• Different reviewing standards than with (expert) developers?
ALSO...

- Keep track of the LLM prompt chaining and logs
- Possibly, using the issue tracker
...AND WE SHOULD BE AWARE ON HOW OUR STUDENTS CONDUCT LITERATURE REVIEW...
Dear Professor Bavota, professor Di Penta, professor Rocco,

I hope this message finds you well. I am a second-year master's student at the University of XXX, specializing in software engineering. I am writing to request your assistance in locating one of your publications that is relevant to my master's thesis topic.

In my literature review, I came across a citation of your work titled Automated "Graph-Based Integration Testing of Web Applications", which is crucial for my research. However, despite my efforts, I have been unable to find this publication in our university's library or online databases.

All the information I was given about this publication are the following:


I would greatly appreciate it if you could direct me to the source of this publication or share an electronic copy if available. Your guidance and expertise would significantly benefit my research.

Thank you for considering my request. I look forward to your response.

Best regards,
THE PAPER DOES NOT EXIST!
TAKEAWAYS AND CONCLUSIONS
TAKEAWAYS
Developers will write less and less code as the AI support is getting better

→ software engineering education has to adapt to that
TAKEAWAYS

Developers will write less and less code as the AI support is getting better
→ software engineering education has to adapt to that

Instructing SE students on AI applications, Prompt engineering and chaining, but also awareness on AI limitations
**TAKEAWAYS**

Developers will write less and less code as the AI support is getting better
→ software engineering education has to adapt to that

Instructing SE students on AI applications, Prompt engineering and chaining, but also awareness on AI limitations

Need to re-think the development process: A suitable Q/A is required for AI generated software artifacts
Developers will write less and less code as the AI support is getting better
→ software engineering education has to adapt to that

Instructing SE students on AI applications, Prompt engineering and chaining, but also awareness on AI limitations

Need to re-think the development process: A suitable Q/A is required for AI generated software artifacts

Lack of trust and traceability may create consequences also from a legal perspective: still many open challenges in this area
MY (OLD) COURSE GOALS

• Provide advanced software engineering contents mostly focused on software quality
• Stimulate workgroup
• Stimulate investigation on innovative topics
MY (OLD) COURSE GOALS

• Provide advanced software engineering contents mostly focused on software quality
• Stimulate workgroup
• Stimulate investigation on innovative topics

IMPACT ON SE EDUCATION

Favor a responsible, informed integration of LLMs in the software development process
MY (OLD) COURSE GOALS

- Provide advanced software engineering contents mostly focused on software quality
- Stimulate workgroup
- Stimulate investigation on innovative topics

IMPACT ON SE EDUCATION

Favor a responsible, informed integration of LLMs in the software development process
**MY (OLD) COURSE GOALS**

- Provide advanced software engineering contents mostly focused on software quality
- Stimulate workgroup
- Stimulate investigation on innovative topics

**IMPACT ON SE EDUCATION**

Favor a responsible, informed integration of LLMs in the software development process

**TAKEAWAYS**

- Developers will write less and less code as the AI support is getting better → software engineering education has to adapt to that
- Instructing SE students on AI applications, Prompt engineering and chaining, but also awareness on AI limitations
- Need to re-think the development process: A suitable Q/A is required for AI generated software artifacts
- Lack of trust and traceability may create consequences also from a legal perspective: still many open challenges in this area
MY (OLD) COURSE GOALS

- Provide advanced software engineering contents mostly focused on software quality
- Stimulate workgroup
- Stimulate investigation on innovative topics

IMPACT ON SE EDUCATION

Favor a responsible, informed integration of LLMs in the software development process

TAKEAWAYS

Developers will write less and less code as the AI support is getting better → software engineering education has to adapt to that

Instructing SE students on AI applications, Prompt engineering and chaining, but also awareness on AI limitations

Need to re-think the development process: A suitable Q/A is required for AI generated software artifacts

Lack of trust and traceability may create consequences also from a legal perspective: still many open challenges in this area

Questions?

dipenta@unisannio.it
@mdipenta