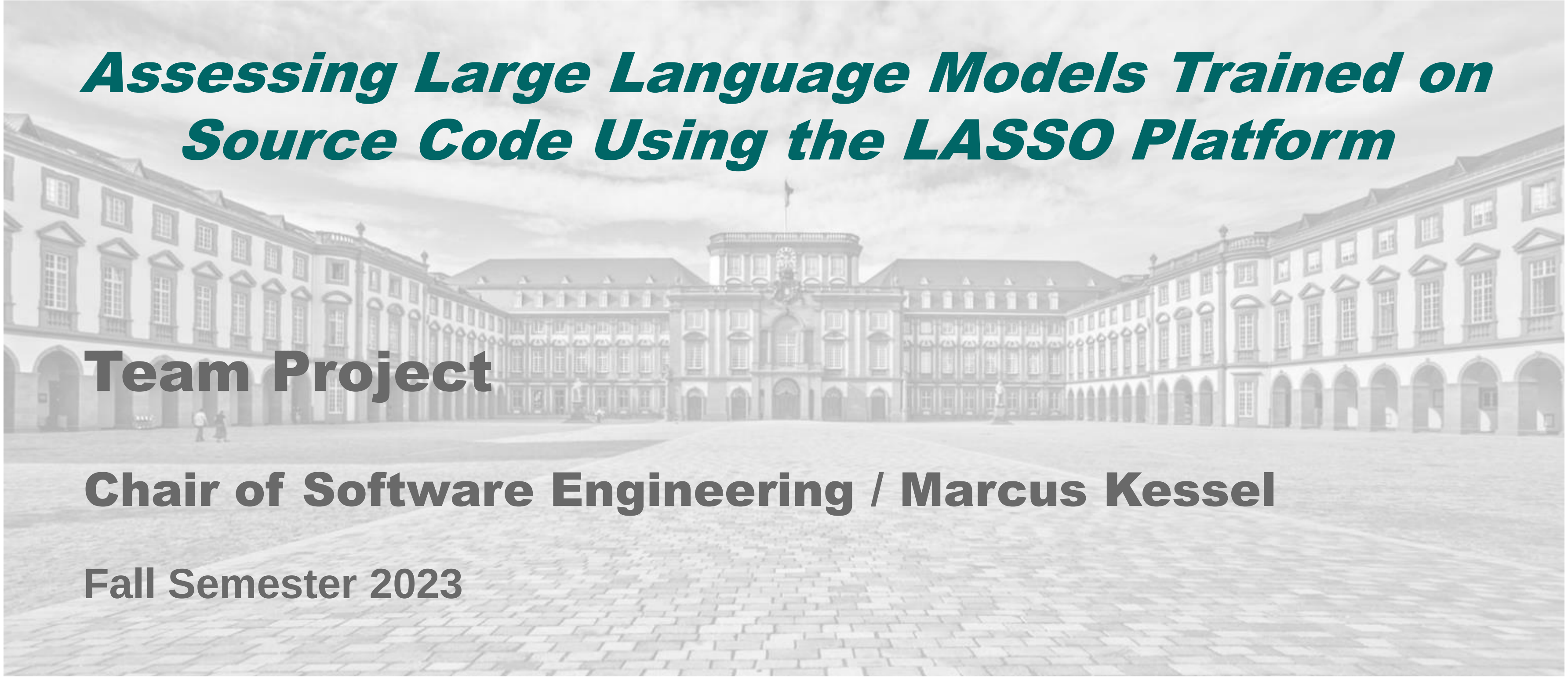


Assessing Large Language Models Trained on Source Code Using the LASSO Platform

Team Project

Chair of Software Engineering / Marcus Kessel

Fall Semester 2023

UNIVERSITY
OF MANNHEIM
School of Business Informatics
and Mathematics

LLMs (and Chatbots) for Source Code

Trained on massive amounts of open source code

OpenAI

ChatGPT

Codex

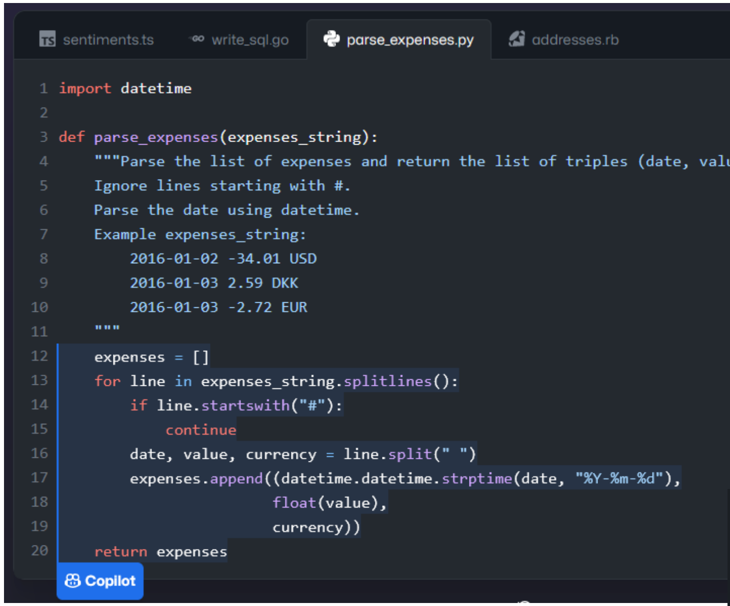
GitHub Copilot

tabnine

StarCoder

… many more

Code & Test Generation Tasks



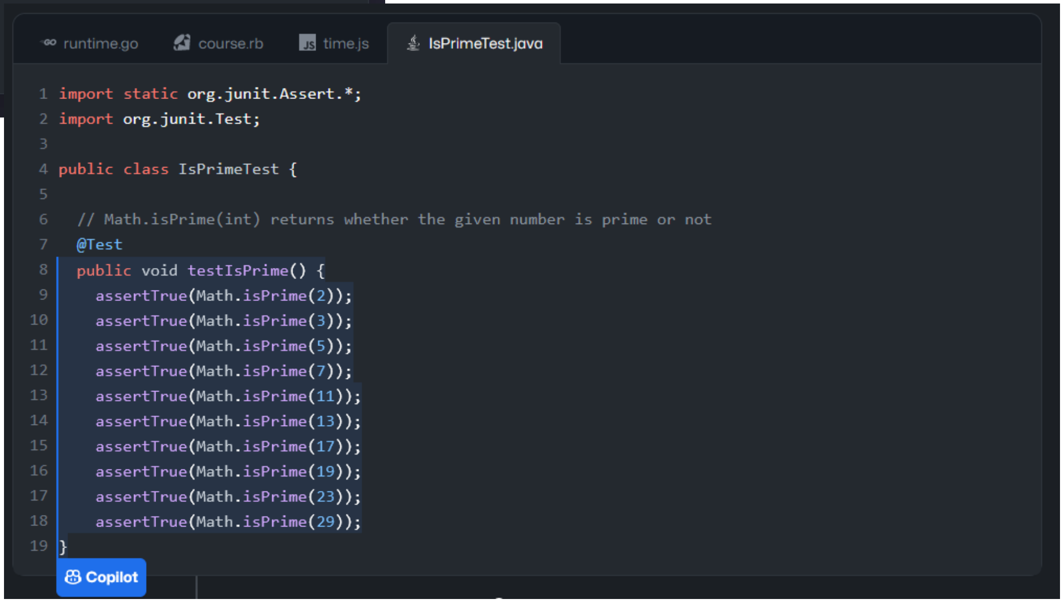
```python
import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date, value, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2016-01-02 -34.01 USD
        2016-01-03 2.59 DKK
        2016-01-03 -2.72 EUR
    """
    expenses = []
    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
                        float(value),
                        currency))
    return expenses
```

Code Generation
(Program Synthesis)

GitHub Copilot

Test Generation
(Inputs and Outputs)



```java
import static org.junit.Assert.*;
import org.junit.Test;

public class IsPrimeTest {

  // Math.isPrime(int) returns whether the given number is prime or not
  @Test
  public void testIsPrime() {
    assertTrue(Math.isPrime(2));
    assertTrue(Math.isPrime(3));
    assertTrue(Math.isPrime(5));
    assertTrue(Math.isPrime(7));
    assertTrue(Math.isPrime(11));
    assertTrue(Math.isPrime(13));
    assertTrue(Math.isPrime(17));
    assertTrue(Math.isPrime(19));
    assertTrue(Math.isPrime(23));
    assertTrue(Math.isPrime(29));
}
```

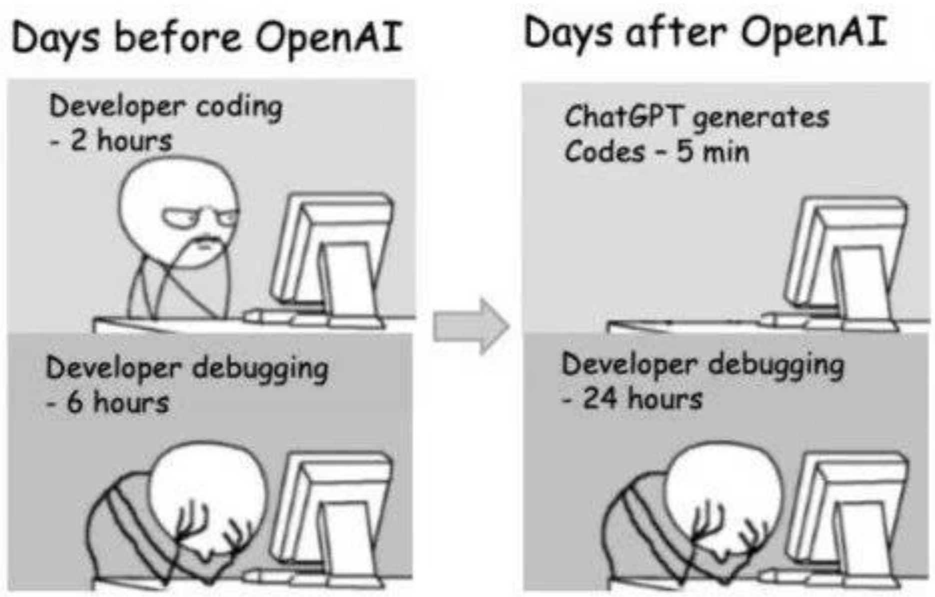
Jack of all Trades, Master of None (?)

"… the robots are coming …"

"… they replace developers …"

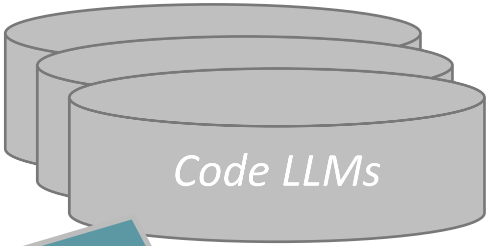
*"… hallucinating bullsh** …"*

"… frees from boring tasks …"


Days before OpenAI
Developer coding - 2 hours
Developer debugging - 6 hours
Days after OpenAI
ChatGPT generates Codes - 5 min
Developer debugging - 24 hours

/r/ProgrammerHumor

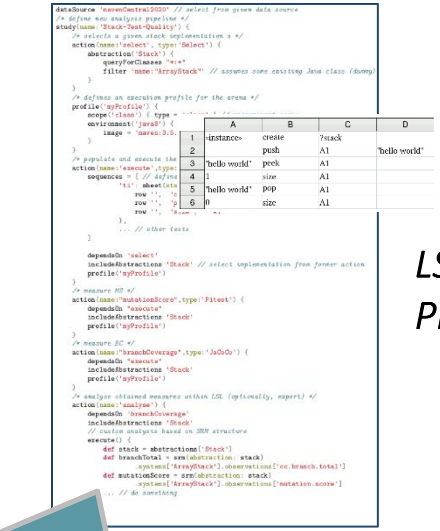
Assessing Code LLMs



Measure and Analyze Functional/Non-Functional Behaviour

Execution-based Program Analysis at Scale

<u>*State-of-the-Art Benchmarks*</u>
Curated Coding Task Collections
→ *HumanEval, MBPP, MultiPL-E* etc.



```
dataSource 'mavenCentral2020' // select from given data source
/* define new analysis pipeline */
study(name:'Stack-Test-Quality') {
    /* selects a given stack implementation s */
    action(name:'select', type:'Select') {
        abstraction('Stack') {
            queryForClasses "*:*"
            filter 'name:"ArrayStack"' // assumes some existing Java class (dummy)
        }
    }
    /* defines an execution profile for the arena */
    profile('myProfile') {
        scope('class') { type =
        environment('java8') {
            image = 'maven:3.5.
        }
    }
    /* populate and execute the
    action(name:'execute',type:
        sequences = [ // define
                'ti': sheet(sta
                    row '',  'c
                    row '',  'p
                    row '',  'size',
                ),
                ... // other tests
        ]

        dependsOn 'select'
        includeAbstractions 'Stack' // select implementation from former action
        profile('myProfile')
    }
    /* measure MS */
    action(name:"mutationScore",type:'Pitest') {
        dependsOn "execute"
        includeAbstractions 'Stack'
        profile('myProfile')
    }
    /* measure BC */
    action(name:"branchCoverage",type:'JaCoCo') {
        dependsOn "execute"
        includeAbstractions 'Stack'
        profile('myProfile')
    }
    /* analyse obtained measures within LSL (optionally, export) */
    action(name:'analyse') {
        dependsOn 'branchCoverage'
        includeAbstractions 'Stack'
        // custom analysis based on SRM structure
        execute() {
            def stack = abstractions['Stack']
            def branchTotal = srm(abstraction: stack)
                    .systems['ArrayStack'].observations['cc.branch.total']
            def mutationScore = srm(abstraction: stack)
                    .systems['ArrayStack'].observations['mutation.score']
            ... // do something
```

	A	B	C	D
1	«instance»	create	?stack	
2		push	A1	"hello world"
3	"hello world"	peek	A1	
4	1	size	A1	
5	"hello world"	pop	A1	
6	0	size	A1	

LSL Analysis Pipelines

Delivers expected functionality?
High-quality code?
Better than Code Search?
Code Attribution?
Risks?

Goal (1)

- LASSO is a leading edge software observatorium that allows advanced search and analysis techniques to be applied to “big code”. Among other things, this simplifies experimentation and the validation of tools and software engineering approaches.
- The goal of this team project is to study leading-edge code LLMs (focus: code and test generation tasks) with the help of the LASSO platform. This includes –
  - Integrating state-of-the-art Code LLMs into LASSO to enable comparisons
  - Integrating/running established benchmarks (coding tasks)
  - Setting up analysis pipelines in LASSO’s scripting language, LSL, to automate the experimentation process including –
    - Test-Driven Assessment of Functional Behaviour
    - Code Quality Measurements
    - Code Attribution (code shared with original data)
    - Comparison with Code Search / Recommendation

Goal (2)

- Participants
  - 6 students
- Length
  - 6 months
- Prerequisites
  - (Java) Programming
  - Fundamental understanding in machine learning
- Language
  - English
- Organisation
  - Goals and timetable defined by agreement with the supervisor
- Applicable to MMDS: yes
- Online: By agreement

- Supervisor
  - Marcus Kessel