Learning Deep Semantics for Test Completion

Pengyu Nie, Rahul Banerjee, Junyi Jessy Li, Raymond J. Mooney, Milos Gligoric

TEXAS
The University of Texas at Austin

ICSE 2023

partially supported by NSF
CCF-1652517
CCF-2107291
IIS-2145479
CCF-221769

Motivation: Writing Tests is Tedious

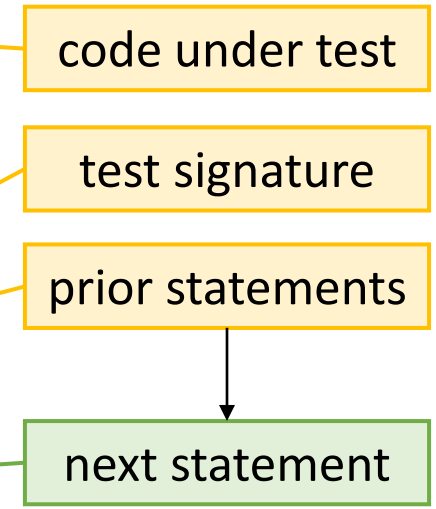
- Testing is the most frequently-used technique to ensure software correctness
- Writing tests can take a lot of manual efforts (~50% of development time)
- Automatically generated tests (e.g., by random testing) have stylistic issues and do not replace the need of manual efforts

Goal: developing ML models to assist developers in writing tests

Task: Test Completion

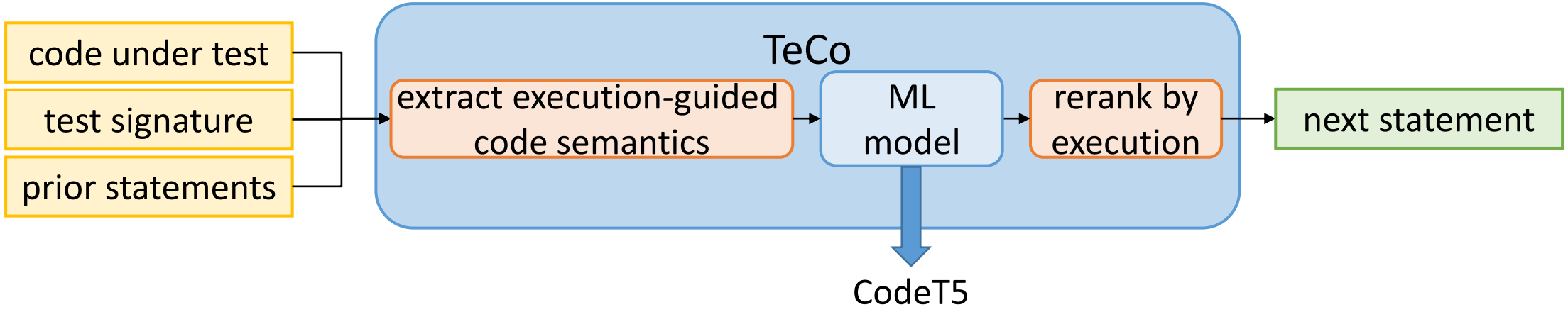
- Complete one statement at a time

```
public class GMOperation extends org.im4java.core.GMOperation {
  public GMOperation addImage(final File file) {
    if (file == null) {
      throw new IllegalArgumentException("file must be defined");
    }
    getCmdArgs().add(file.getPath());
    return this;
  }
  ...
}
public class GMOperationTest {
  @Test
  public void addImage_ThrowsException_WhenFileIsNull() throws Exception {
    exception.expect(IllegalArgumentException.class);
    ...
  }
  ...
}
```



Example from project sharneng/gm4java, GMOperationTest.java

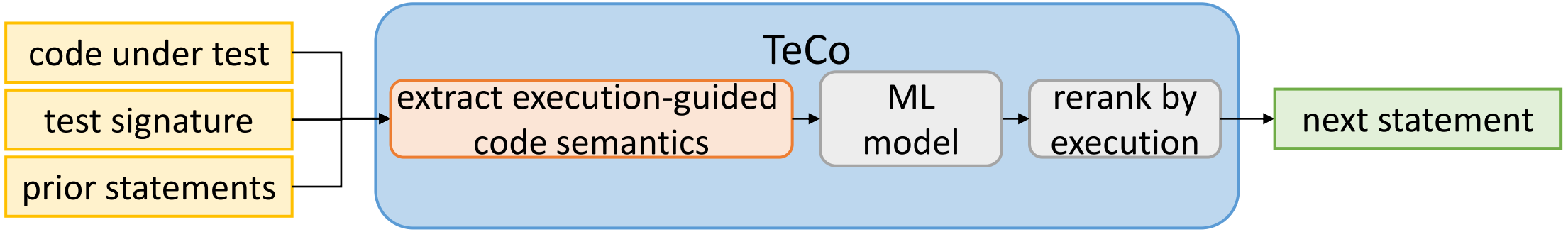
TeCo: ML + Execution for Test Completion



- Test completion can greatly benefit from reasoning about execution
  - types, program state (local and global), callable methods, etc.
  - whether the output is executable
- TeCo uses code semantics as inputs and performs reranking by test execution

Wang et al., CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In EMNLP'21.

Execution-Guided Code Semantics



- Execution results: program state after executing prior statements

S1 local var types | **S2** absent types | **S3** uninitialized fields

- Execution context: code fragments relevant for predicting next statement

S4 setup teardown | **S5** last called method | **S6** similar statement

Execution-Guided Code Semantics: Example

```
public class GMOperation extends org.im4java.core.GMOperation {
  public GMOperation addImage(final File file) {...}
... }

public class GMOperationTest {
  GMOperation sut;
  @Before public void setup() { ... sut = new GMOperation(); ... }
  @Test
  public void addImage_ThrowsException_WhenFileIsNull() throws Exception {
    exception.expect(IllegalArgumentException.class);
                ?
  }
... }
```

S2 absent types

types that are required by the code under test, but are not available before executing the next statement

S4 setup teardown

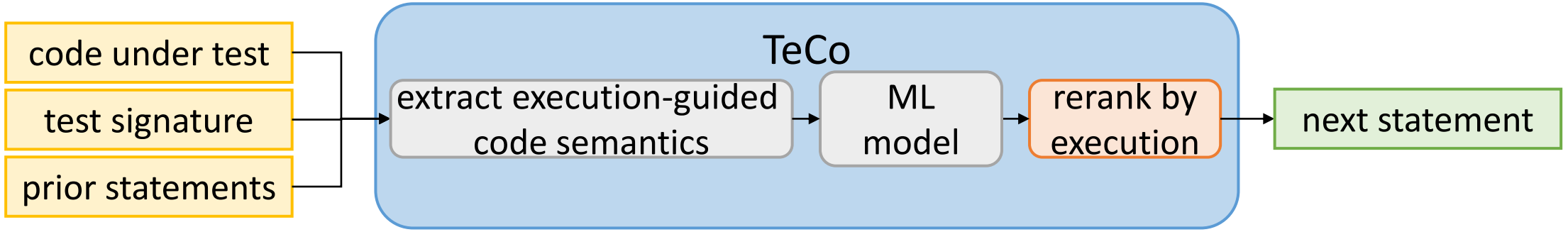
methods executed before/after the test by the testing framework

CodeT5 prediction `new GMOperation().addImage(null);` ✗

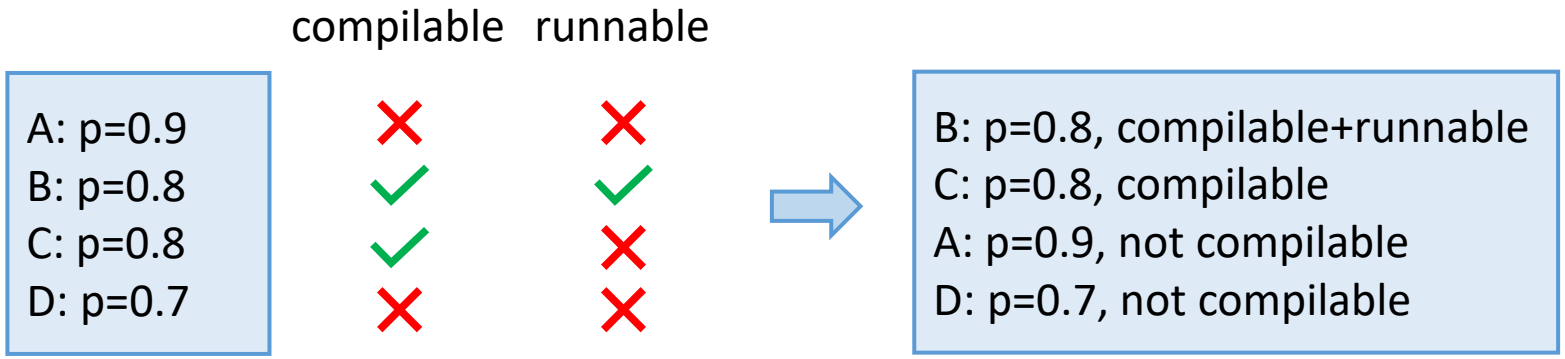
compilation error: addImage is overloaded addImage(File); addImage(Object)

TeCo prediction `sut.addImage((File) null);` ✓

Reranking by Execution



- Reranking: prioritize generating compilable and runnable statements



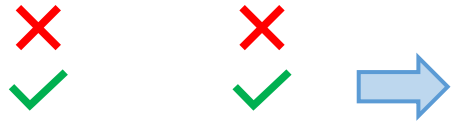
Reranking by Execution: Example

```
public class GMOperation extends org.im4java.core.GMOperation {
  public GMOperation addImage(final File file) {...}
... }

public class GMOperationTest {
  GMOperation sut;
  @Before public void setup() { ... sut = new GMOperation(); ... }
  @Test
  public void addImage_ThrowsException_WhenFileIsNull() throws Exception {
    exception.expect(IllegalArgumentException.class);
                  ?
  }
... }
```

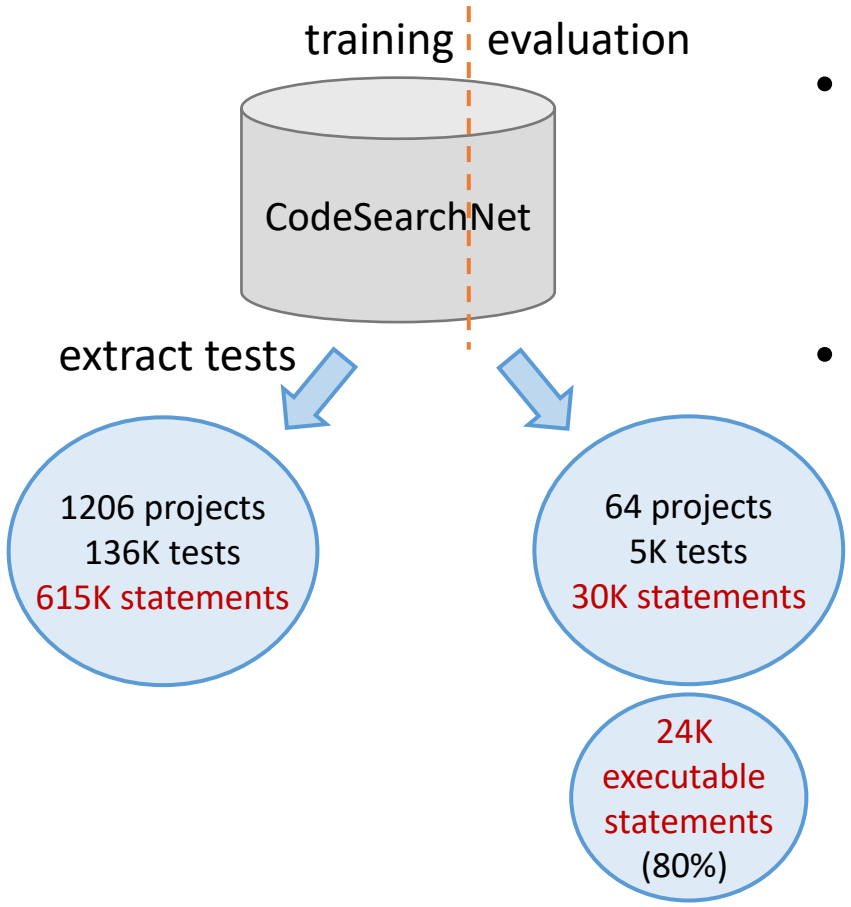
compilable runnable

```
sut.addImage(null);
sut.addImage((File) null);
...
```



```
sut.addImage((File) null);
sut.addImage(null);
...
```

Evaluation: Dataset

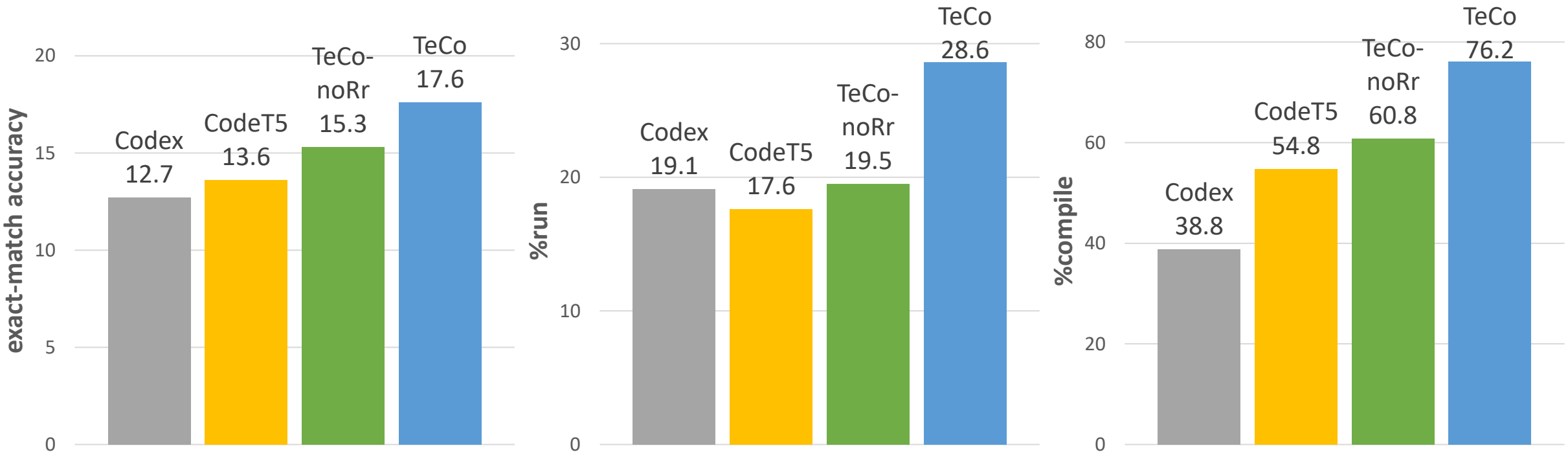


- Developer-written tests from open-source Java projects in CodeSearchNet
  - same dataset and split as used in pre-training CodeT5
- 80% of the evaluation set statements are executable
  - computing additional metrics on the executability of the output statements

Evaluation: Setup

- Metrics
  - syntax-level correctness: exact match accuracy (similarity-based metrics in paper)
  - functional correctness: %run, %compile
- Baselines
  - Codex: 175B model pre-trained on GitHub (Mar 2023)
  - CodeT5: 220M model pre-trained on CodeSearchNet, fine-tuned on our dataset
- Models
  - TeCo-noRr: code semantics + CodeT5
  - TeCo: code semantics + CodeT5 + reranking by execution
- Configurations
  - 4x Nvidia 1080Ti GPUs, Linux
  - run each experiment three times with different random seeds

Evaluation: Test Completion



TeCo improves the accuracy of test completion by 29%, and is better in generating compilable/runnable test statements

Conclusions

- TeCo: ML + execution model for test completion
- The use of code semantics and reranking by execution is important for increasing ML models' performance on test completion
- Dataset: 1,270 projects, 131K tests, 645K statements

https://github.com/EngineeringSoftware/teco

Pengyu Nie, Rahul Banerjee, Junyi Jessy Li, Raymond J. Mooney, Milos Gligoric

pynie@utexas.edu

joining Waterloo CS as assistant professor starting from 9/1