ML-Based Test Prioritization “to-fail-first” as a Service

Alexandre Vernotte  Aymeric Cretin

13/09/2022
Outline

- Test case prioritization: state of the art
- Fail-first ML based prioritization
- ML experimentation results
- Implementation
- Conclusion
Execute test cases in an order that satisfies a prioritization objective:
- Business requirements criticality
- Usage patterns frequency
- Test case failure probability (Fail First)
- ...

*Fail first* TCP (Test Case Prioritization) aims at executing failing test cases as early as possible
- Faster bug discoveries means faster bug fixes
- Combined with test selection, reduces regression test costs (time & resources)
Test Case Prioritization: State of the Art

Many Fail first TCP techniques were created over the last 20 years:

- Prioritization by promoting test cases diversity
  - E.g., compute string distance between test cases

- Prioritization by predicting the test cases’ result, which may rely on:
  - Code coverage
  - Code changes
  - Past verdicts
  - Code & test complexity, customer-assigned costs, severity of detected faults, etc.
Test Case Prioritization: State of the Art

Recently, ML (Machine Learning) approaches emerged to tackle TCP
- Fail first TCP is a verdict prediction problem = typical ML problem
- Capability to combine multiple data sources (e.g. code coverage data + historical data)

Indication that ML may outperform heuristics

A wide variety of ML models
- Deep Neural Networks
- Genetic Algorithms
- Decision Trees
- Reinforcement Learning
Fail-first ML based TCP Architecture

CI/CD Server

CI/CD cycle

1. Build
2. Prioritize
3. Test
4. Notify
5. Deploy
6. Extract

Run cycle

Test Suite

Prioritized Test Suite

ML Model

Features extraction

Build data

Test Data Repository

Prioritization module

Updates

VCS (Git, SVN)

Push code

Get source code

Testing of Trustworthy Systems
Test and code features for TCP

History-based features
- Last N verdicts (N with range 4 – 10)
- Execution time (mean of the last 3 runs)
- Execution frequency
- Time (number of CI/CD cycles) since last execution

Test case related features
- Age of the test case
- Number of test methods
- Whether the test case was modified
- Text similarity score with modified source code files
Models Experimentation

Experimentation were conducted on 2 classes of ML model:
- Decision Trees (DTs)
- Reinforcement Learning (RL)

The models were evaluated on 13 software development projects:
- 12 GitHub projects obtained from the RTPTorrent dataset
- 1 live product (Smartesting Yest)

Results consistently showed that DTs are superior to RL models (w.r.t. the feature set)
- Much faster to train (seconds for DTs, several hours for RL at best)
- Better prediction scores (APFD – Average Percentage of Faults Detected)
Experimentation Results: DTs vs baseline

- deeplearning4j
- DSpace
- yest
- LittleProxy
- dynjs
- buck
Experimentation Results: DTs vs RL
Implementation: Comet API

Comet API
- Online prioritization requests
- Resources management:
  - Projects
  - Test cycles
  - Tests
  - Test features
- Can be easily integrated to a CI server or a test management tool
Implementation: CI integration

- CI integration
  - Targets automated tests
  - Jenkins Plugin

- Collected data
  - Build info
  - Tests features
  - Tests results
Implementation: Test management Tool

Test management tool integration

- Targets automated and manual test
- Java and Python clients
Remaining Barriers for ML-driven TCP

Major features that are too cumbersome to compute
- Per test code coverage can rarely be obtained without hassle
- NLP may be an acceptable lightweight alternative to per test code coverage


Defining an explicit testcase execution ordering ➞ A lot harder that it appears to be!
- Most testing tools do not allow this (has to do with test cases having to be independents)
- Multi-module (e.g., maven) projects add another layer of complications
Conclusions

- Regression tests are time-consuming to run, and TCP can help reduce that cost.
- SoTA implies that ML models outperform heuristics.
- Experimentation suggests that decision trees yield better results than RL models.
- Comet is a fail-first TCP API that can integrate CI/CD processes.
- Comet can also integrate test management tools to prioritize manual tests.
Any further questions?

alexandre.vernotte@femto-st.fr

aymeric.cretin@smartesting.com