Test Case Prioritization Optimization with Machine Learning

A case-study at BNP Paribas

João Lousada

Supervisors: Prof. Doutor Rui Dilão
Dr. Miguel Ribeiro

BNP PARIBAS

TÉCNICO LISBOA

01

Introduction

Problem Definition

02

State of the Art

The starting point

03

Study

Novel Approach

04

Conclusions

Limitations and Future Work

Introduction

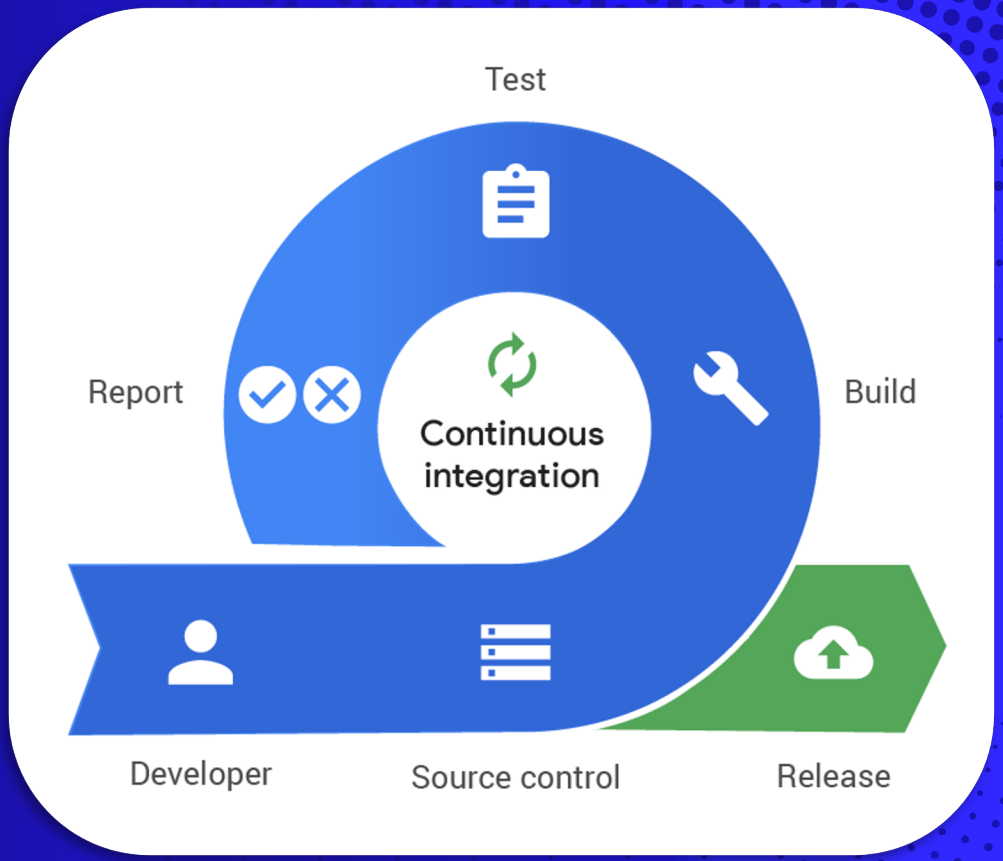
Modern Software Systems

Continuous Integration (CI)

Commit

Regression Testing

Merge



Definition of the Problem

Timestamp		Test A	Test B	Test C	Test D	Author	
9:30	Commit 1	✓	✓	✓	✓	P.	
10:00	Commit 2	✓	✓	✓	✓	M.	
11:00	Commit 3	✓	X	✓	X	F.	*Regression*
11:20	Commit 4	✓	✓	✓	✓	F.	*Progression*
15:00	Commit 5	✓	✓	✓	✓	P.	

- Stable - Unstable

Scenario 1	Scenario 2	Real-World
5 Commits/day 4 Tests 3 Developers	24 Commit/day 20 Tests 10 Developers	100 Commits/day +3.000 Tests 90 Developers
Time Spent:	Time Spent:	Time Spent:
40 min	16h	1.2 yr

Considering each test takes 2 minutes to run.

100 Commits/day
+3.000 Tests
2min/Test
90 Developers

1.2 Years of Testing Time
for 1 day of commits

Basic Solutions

- Less Commit Frequency ?
- Less Tests ?
- More Computer power ?
- Manual testing ?

These solutions do not scale.

- Poor-quality Software
- Higher Risk/Lower Correctness
- Significant Increase in Costs
- Lower Productivity

Objectives

Savings

Time and Resources

Lag-Time

Reduce the time between commit and feedback

Human

Boost Productivity and Confidence

What can be optimized?

01

CI Configuration

Adopt different strategies

02

Regression Testing

Test “smarter”

01 - CI Configuration

Naïve

Run every test for every single commit.

Goggle

Pre-Commit and Post-Commit Testing.

Uber

Always Green Mainline

BNPP (Simplified)

All test cases are run on a loop, without any specific test ordering.

Once a regression is found binary-search algorithm finds first failing commit.

800 cores.
Random Testing.
Large Lag-Time.

02 – Regression Testing

Which tests are more relevant ?

Test Case Prioritization (TCP)

Given:

T: Set of Tests

PT: Permutations of T

$f: PT \rightarrow \mathbb{R}$

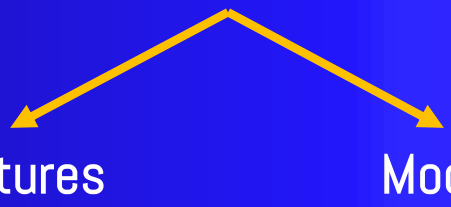
Find a subset ***T' such that:***

$$[f(T') \geq f(T'')], \forall T'' \in PT$$

f can be a measure of early fault detection.

Machine Learning

How do we learn from data?



Most common Features

Coverage-based

Functions, Methods, Lines of Code, Files Mofidied,...

History-based

Execution history, failure history, Last execution time, ...

Similarity-based

Hamming Distance between two tests, number of similar method calls, ...

Others

Author, Experience, Test Duration, Comments, ...

Main ML Algorithms for TCP

Supervised Learning

Logistic Regression, Decision Trees, Random Forests, Support Vector Machines

Unsupervised Learning

Clustering

Reinforcement Learning

Learning by trial-and-error by receiving feedback.

Deep Learning

Restricted Boltzmann Machines, Deep Belief Networks, Artificial Neural Networks.

Types of Predictions

Commit Fails

Given a commit, predict pass or fail every test.

Test Fails

Predict likelihood of test fail, in a given commit.

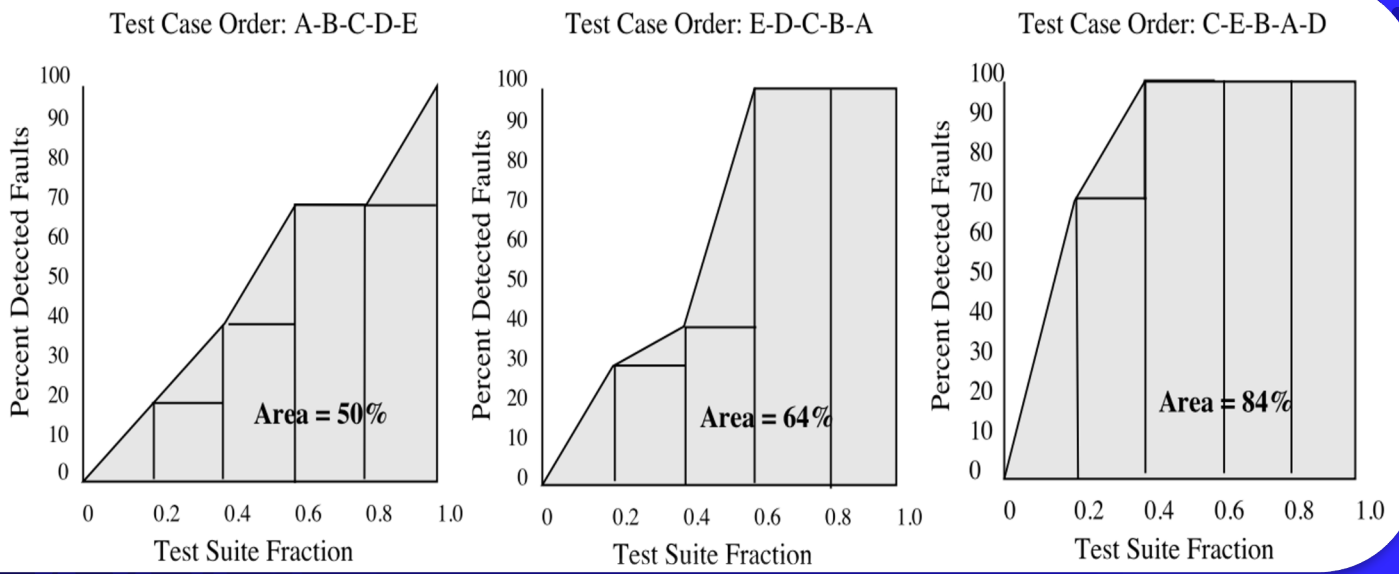
Test Transitions

Predict likelihood of test transitition. Regression/progression

APFD Metric

test	fault									
	1	2	3	4	5	6	7	8	9	10
A	x				x					
B	x				x	x	x			
C	x	x	x	x	x	x	x			
D					x					
E								x	x	x

$$\text{APFD} = 1 - \frac{TF_1 + \cdots + TF_n}{nm} + \frac{1}{2n}.$$



Good test orderings maximize the AUC.

Average Percentage of Transition Detection (APTD)

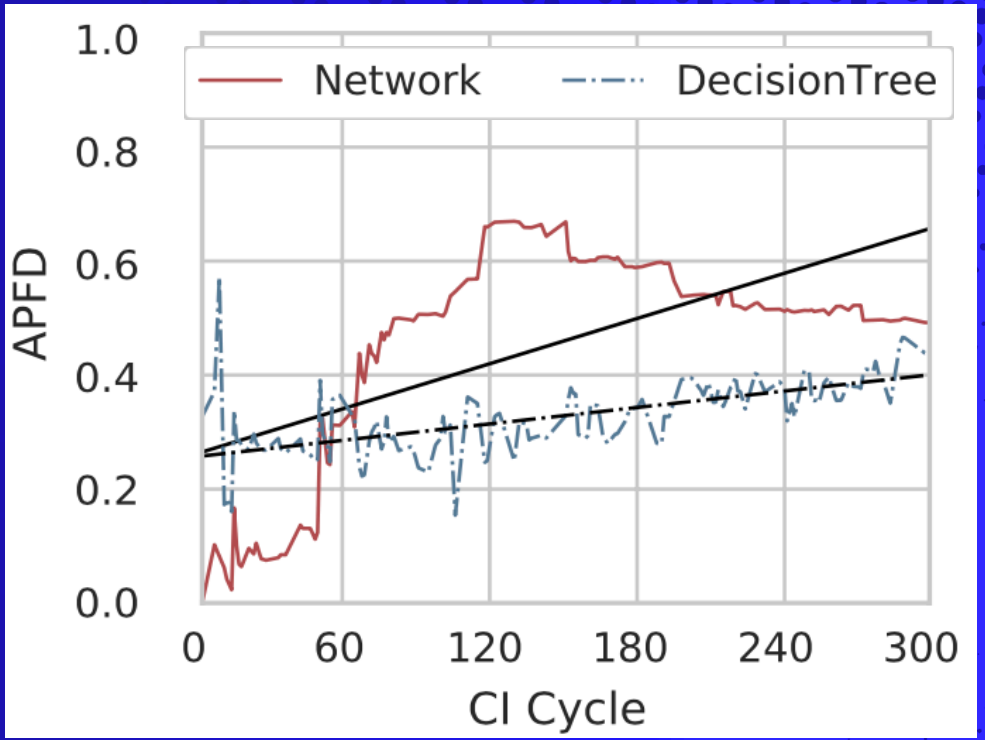
Proposed Solution – I

Reinforcement Learning (RL)

RL handles problems that involve learning which course of action to take.

Features:

- Test Duration
- Last Execution
- Failing History



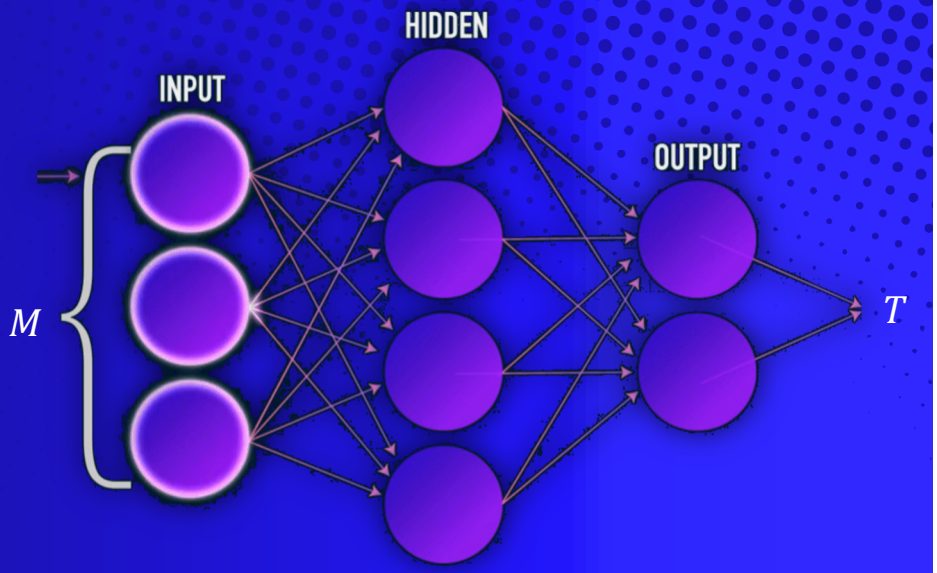
Proposed Solution – II

Find a mapping between modified files (M) and tests that suffered transitions (T).

$$f: M \rightarrow T$$

Input: M – Vector of Files Modified
Model: f – ML Algorithm
Output: $\mathbf{T}$ – Vector of Tests

ANN



Problem

Too many inputs and outputs!

Neural Network Embeddings

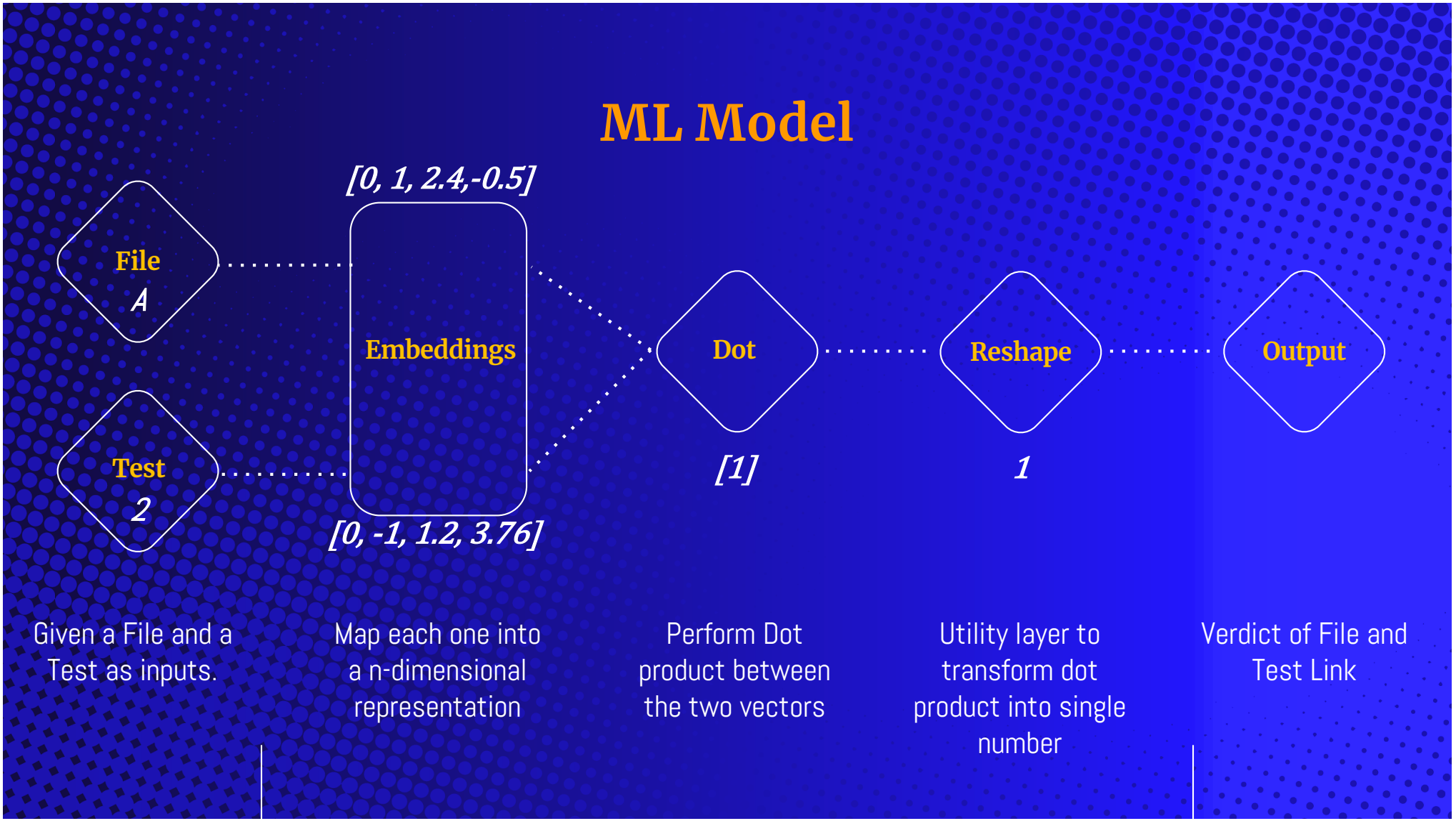
Embeddings

File A = [0, 1, 2.4,-0.5]
Test 2 = [0, -1, 1.2, 3.76]

Task

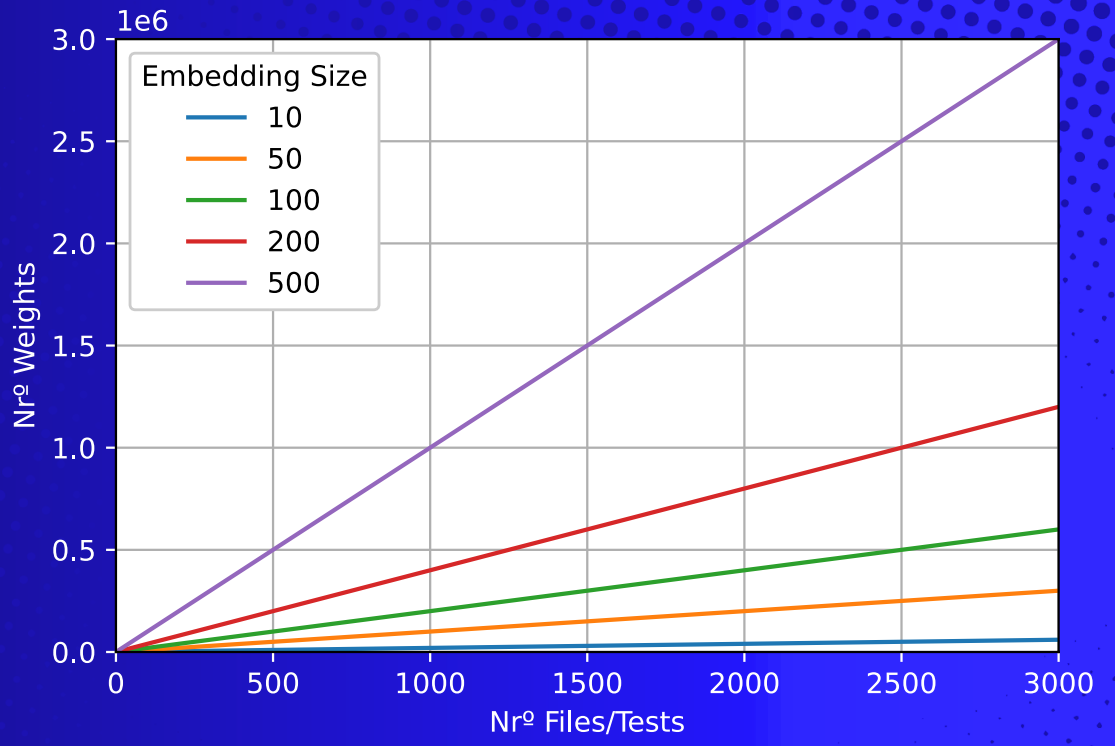
(file, test) → *link*

Clean Data → Generate Training Set → Train Model → Predict → Visualize


ML Model
[0, 1, 2.4,-0.5]
File
A
Test
2
Embeddings
[0, -1, 1.2, 3.76]
Dot
[1]
Reshape
1
Output
Given a File and a Test as inputs.
Map each one into a n-dimensional representation
Perform Dot product between the two vectors
Utility layer to transform dot product into single number
Verdict of File and Test Link

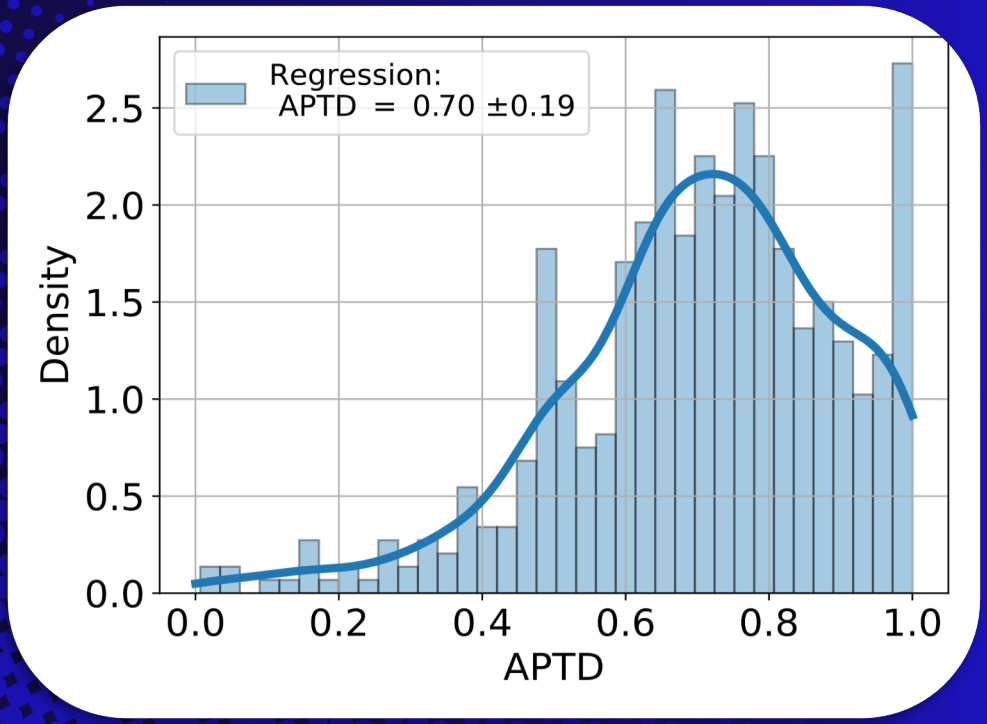
Number of weights in the model

Parameters	Typical Values
Number of Files	~1.000
Number of Tests	-3.000
Embedding Size	[50, 100, 200]
Negative Ratio	[1, 2, 3, 4]
Batch-Size	[1, 5, 10]
Epochs	[10, 100]
Task	Classification or Regression



Number of weights grows linearly with number of files/tests and embedding size.

Results



Histogram with Kernel Density Plot

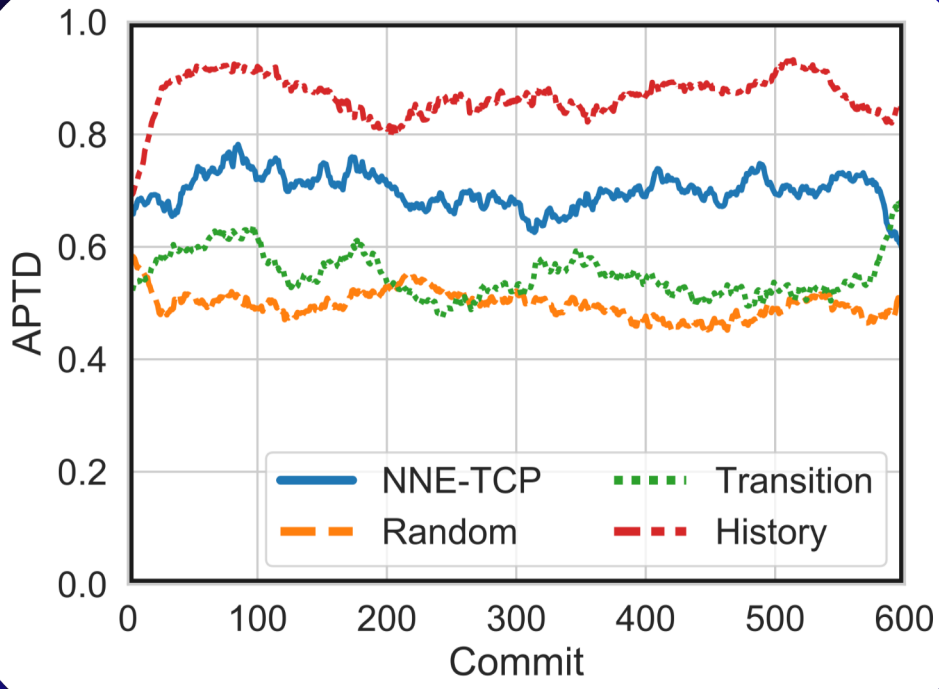
Average Percentage of Transition Detection (APTD)

0.70

Mean APTD

Noise

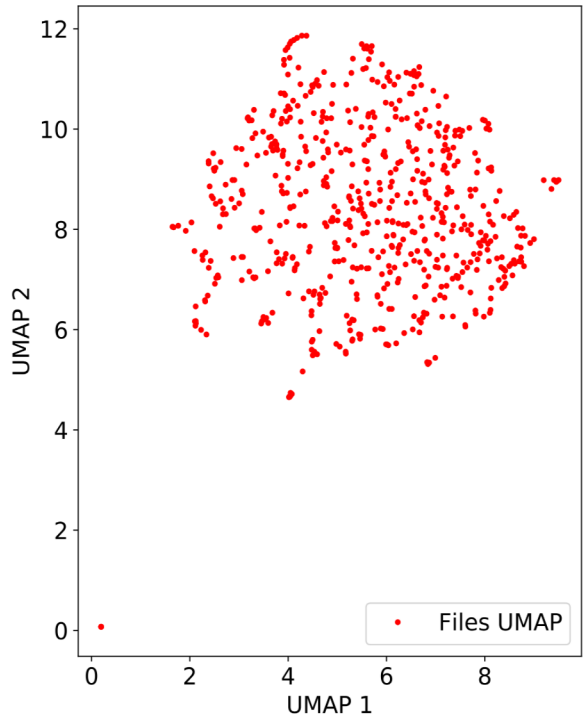
Comparison with Traditional Methods

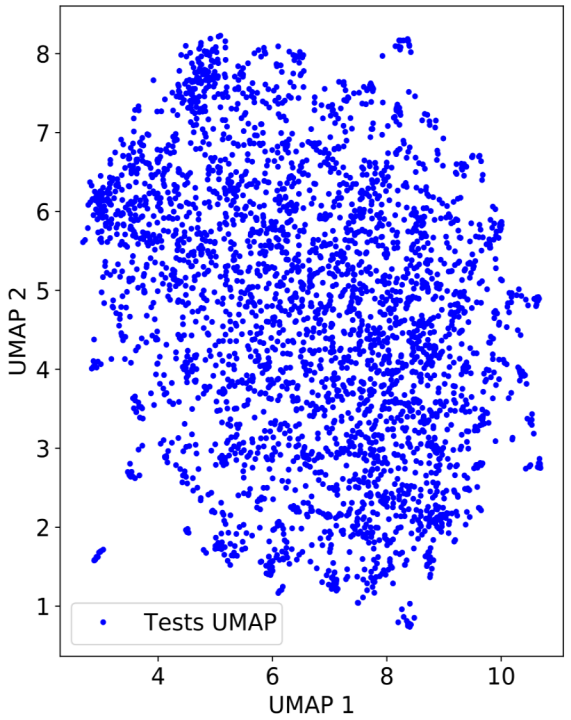


- Random - Random Prioritization.
- Transition - Fixed test ordering by Probability of Transition.
- History - Tests that transitioned recently have higher ranking.

Method	APTD
NNE-TCP	**0.70**
Random	0.50
Transition	0.59
History	0.87

Entity Visualization





Conclusions

01 Two Solutions

Reinforcement Learning
Neural Network Embeddings

02 Improvement

Significant improvements in comparison to current approach

03 Traditional Methods

Better than Random and Transition. Worse than History.

04 Visualization

Visualization is a valuable tool to find structure in the data.

Future Work

01 Parameter Tuning

Find optimal set of parameters

02 More Features

Combine file-test linkage with History method or RL.

03 Collect More Data

Machine Learning models require vast amounts of data.

04 Comparison to other ML Algorithms

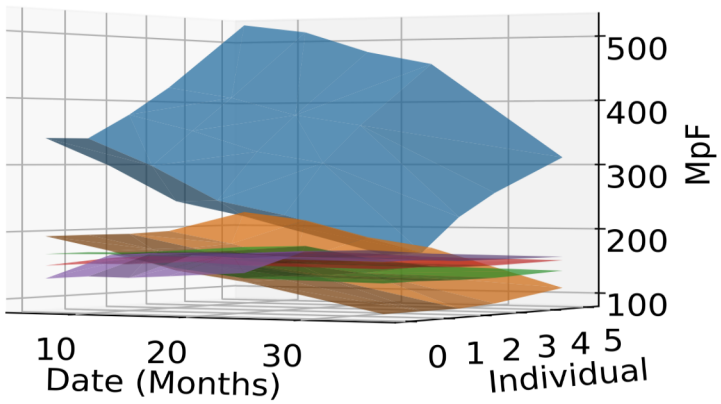
Main References

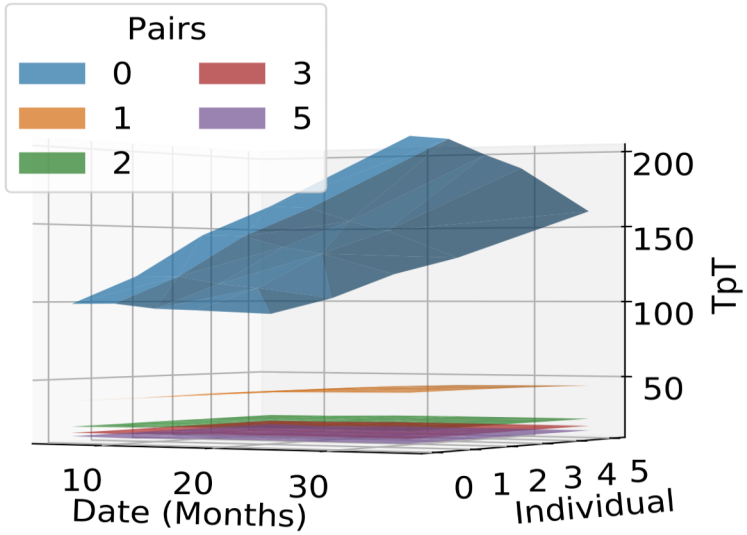
- V. H. S. Durelli et al., Machine learning applied to software testing: A systematic mapping study. *IEEE Transactions on Reliability*
- Y. Shin. *Extending the Boundaries in Regression Testing: Complexity, Latency, and Expertise*. PhD thesis, King's College London, 2009.
- D. Marijan, A. Gotlieb, and S. Sen. Test case prioritization for continuous regression testing: An industrial case study.
- B. Busjaeger and T. Xie. Learning for test prioritization: An industrial case study. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineer- ing*
- H. Spieker, A. Gotlieb, D. Marijan, and M. Mossige. Reinforcement learning for automatic test case prioritization and selection in continuous integration. *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*
- G. Rothermel, R. J. Untch, and C. Chu. Prioritizing test cases for regression testing. *IEEE Trans. Softw. Eng.*, 2001.

Thank You!

Questions

Data Cleaning





Date

Remove old files/tests

Individual

Remove unfrequent files/tests

Pairs

Remove unfrequent pairs

Training

K-Fold Cross Validation(CV) .

Allows us to say the model has generalization ability.

Does not overfit our underfit.