Progress Meeting

Evaluating the Trade-offs of Diversity-Based Test Prioritization

Ranim Khojah, Chi Hong Chao
Supervisor: Francisco Gomes de Oliveira Neto

Problem Statement

- Test prioritization - one method to maximize effectiveness with limited resources
- Diversity-based testing(DBT), one of such prioritization techniques, has shown much promise
  - 2 types - Artifact based diversity and Behavioural based diversity
  - Different techniques for both types - String based techniques for a-div, test execution history patterns for b-div
- Each technique has trade-offs - especially when same technique is run on different levels of testing
  - Different trade-offs for different levels of testing
  - Maybe a technique works better on certain level
- Currently literature does not compare diversity techniques on different levels in a holistic manner.
- Knowing how DBT works (or does not work) on system/integration/unit levels will help testers select techniques on certain levels based on context and situation. Understanding trade-offs will open paths to look for alternatives where DBT is suboptimal.

Proposed Solution

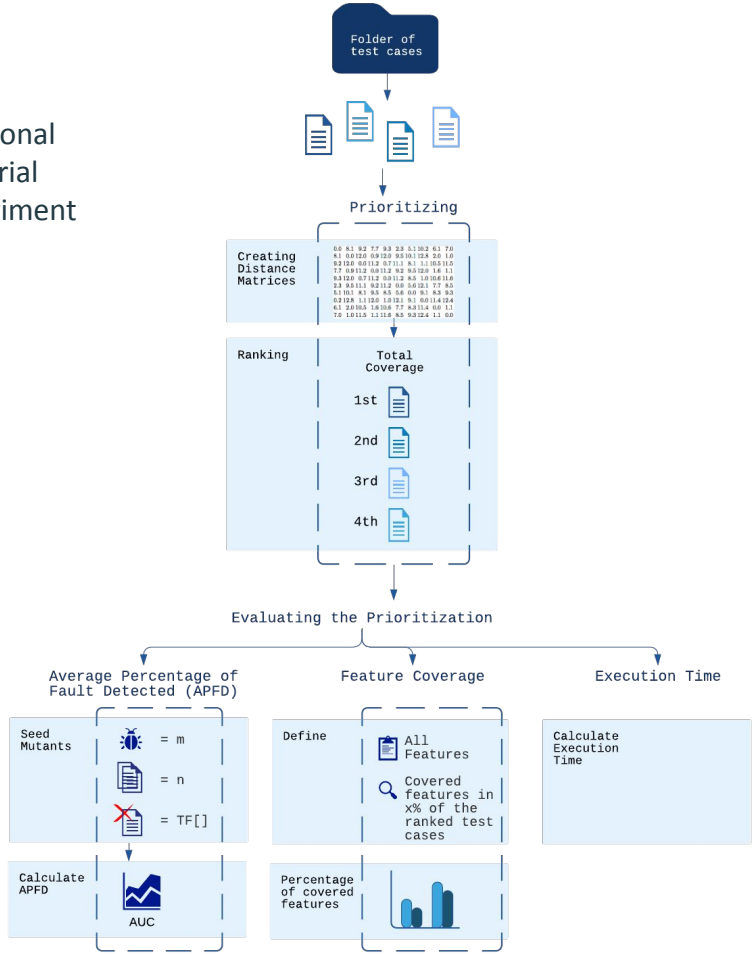
- Create an experiment to uncover cause and effect relationships between different DBTs and levels of testing(LoT)
- Experiment will be a fractional factorial exp. because results of some technique|LoT will not be interesting (prioritized results will be poor, technically infeasible)
- Time taken, coverage, fault detection rates of each technique will be compared on different LoTs.
- Establishing causal relationship will let us have stronger conclusions, and we will know strengths and weaknesses of a technique|LoT pair.
- Techniques used will be relevant in the current literature such that they will be techniques used in the current industry.

Methodology Overview

Artifact-based Diversity	Behavioural-based Diversity
Nat. Lang. Processing	Accuracy
Normalized Compressed Dist	Matthew's Correlation Coefficient
Levenshtein	
Jaccard	

Technique/level	Unit Level	Integration Level	System Level
NLP		X	X
Jaccard Index	X	X	X
NCD	X	X	X
MCC	X	X	
Accuracy	X	X	

- Fractional Factorial Experiment



Defects4J / MultiDistances Current Pipeline for 1 Project

End Goal: Obtain Average Percentage of Faults Detected (APFD) of a-div technique (ex. Jaccard) of an entire project

1. Obtain all (buggy) versions of a project
2. Obtain a list of all test cases (method) names for each version
3. Use test case name list to separate each method of each test class into their own file
4. Input/feed each version's list of test cases (methods) into MultiDistances, specifying technique. Obtain a list of ranked tests.
5. Merge ranked tests together.
    a. Current strategy involves taking the ranked location of the first triggering test in each version and averaging the rank across all the versions
    b. Issue: each version has a different test suite, with different sizes and methods. However, suite is still similar, so could simply be a validity threat, mitigate by saying that isolated faults are more important in a experiment.
6. Get something like this: ----------> APFD $APFD = 1 - \frac{TF_1 + TF_2 + \ldots + TF_m}{nm} + \frac{1}{2n}$

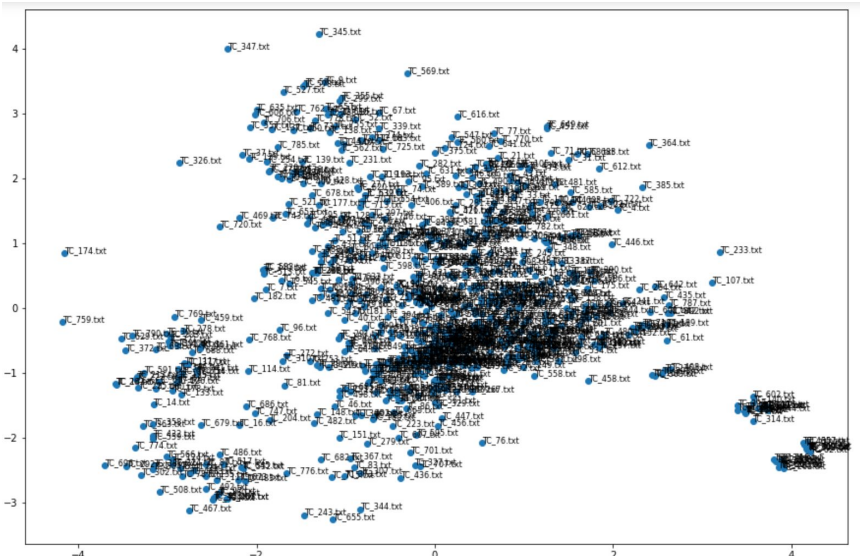
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
TC1				X		X				
TC2		X								
TC3		X							X	
TC4		X	X						X	
TC5			X			X				X
TC6										
TC7	X									
TC8				X				X		
TC9										
TC10					X	X	X	X		

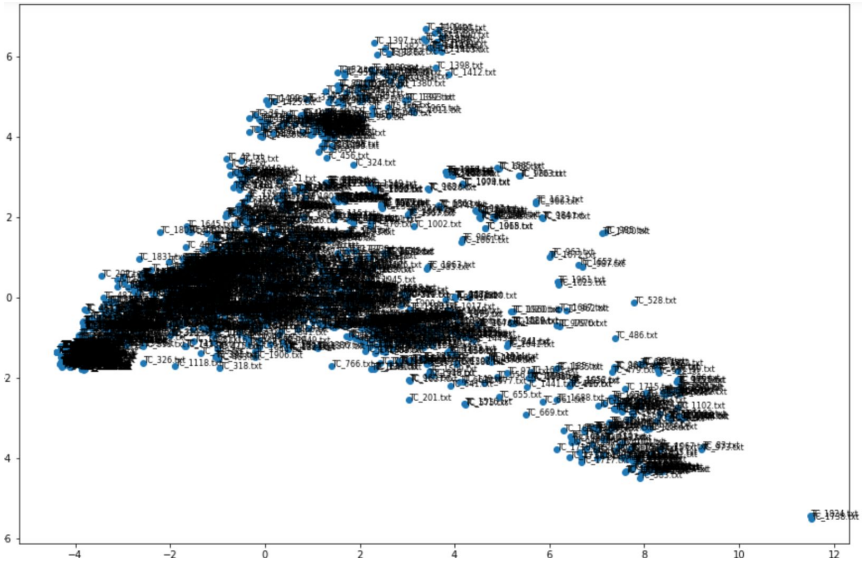
7. Calculate APFD using formula

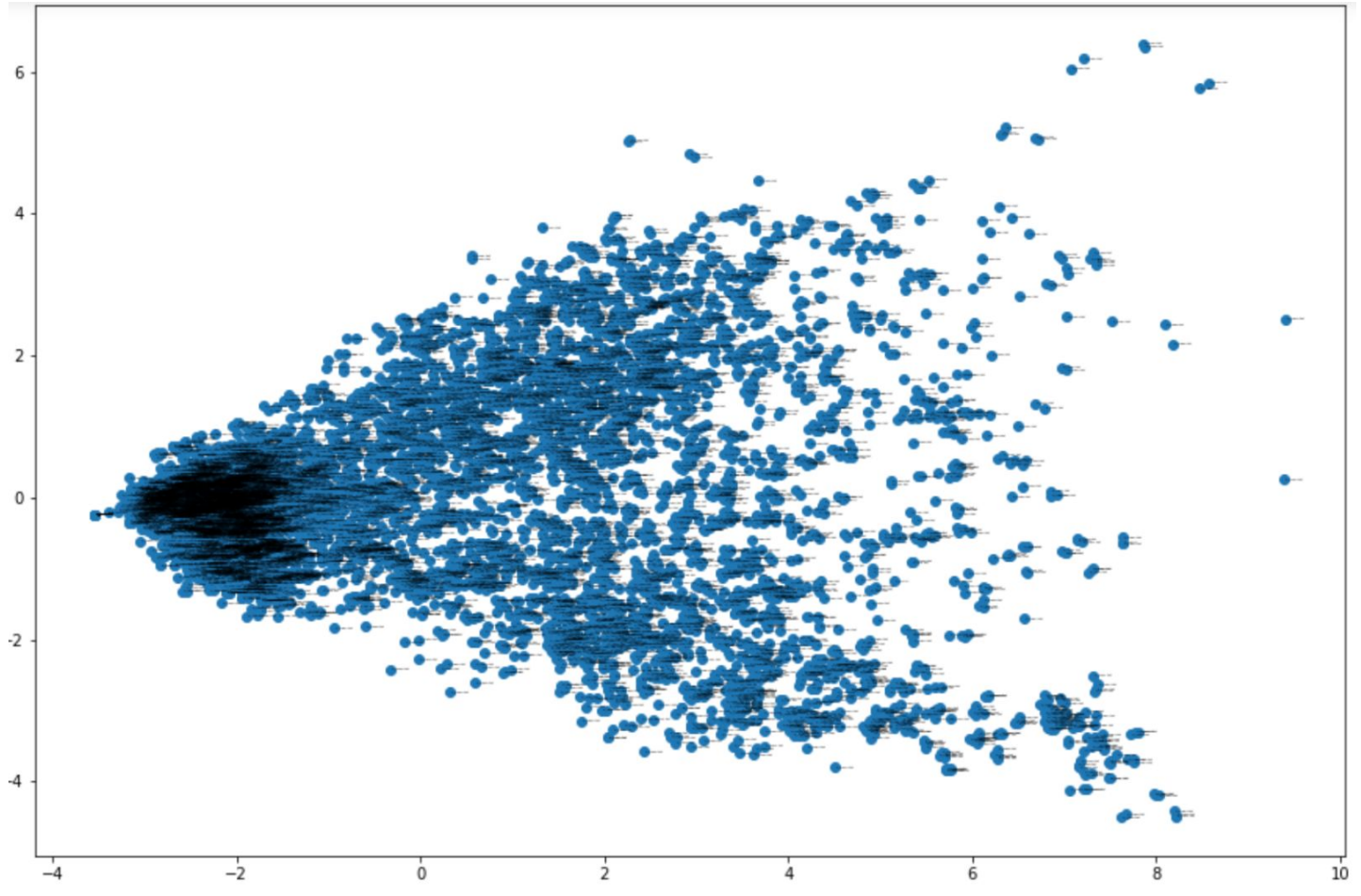
What we have progressed

- Implementation of artefact-based techniques
  - NLP: using a Doc2Vec model
  - Jaccard, Levenshtein, NCD: MultiDistances package.
- Execution on System-level:
  - Execute all artefact-based techniques on Mozilla tests (~200 test cases)
  - Execute NLP on 3 projects from 2 companies' data (see next slides)
  - In terms of requirements/ feature coverage, there was no significant difference between the techniques.
  - In terms of efficiency, NLP: 0.15 sec/1 tc, <<other techniques' exec time needs to be calculated>>
- Execution on Unit-level:
  - Automation of a large majority of processing work through scripts
  - Ranking of test cases for D4j projects
  - A greater understanding of D4j in general

Similarity Maps for two companies' data using NLP






6
4
2
0
-2
-4
−4
−2
0
2
4
6
8
10

What is left to do

- Evaluation of artefact-based on a unit level
  - Fault detection rate - APFD - find a feasible way to merge the results of all ranked tests of each version into 1
    - At the end of the day, a single graph should be produced to easily visualize the APFD results of multiple techniques
  - Coverage - Simple enough with Defects4j, as there is a built in function to do so.
  - Execution time - Difficult - MultiDistances does not seem to have monitoring for how long the process takes. A script is most likely needed to monitor time taken and record execution time. As the ranking implementation of NLP is also taken from Multidistances, one script is most likely enough.
- Evaluation for system-level tests
  - Fault detection: using fault information if available, if not, we need to define a strategy on how to mutate high-level test specifications
  - Execution time - will most likely need a script as mentioned above.
- Behaviour-based diversity
  - NLP and D4j have been taking center stage...
  - MCC
  - Accuracy
- Execution on the Integration level