Research Proposal for Code Smell Algorithm Training Dataset Analysis

Rodger Byrd
rbyrd2@uccs.edu

I. Introduction to the Problem

This proposal is for research involving machine learning and code smells. A lot of previous research has been done on the performance of different algorithms and the detecting of different subsections of code smells. This paper will focus on training datasets with regard to machine learning and code smells. Many different approaches have been taken with regards to training datasets, but they haven't been compared for effectiveness in training the machine learning algorithms.

A code smell is a description of a pattern in code that might cause deeper problems. Code smells are not easily found and can vary by programming language and development methodologies. The existence of code smells and antipatterns implies that there are potential problems with software sustainment and imply there are issues with the design. Kent Beck coined the term "code smell" [1] and defined as "certain structures in the code that suggest (sometimes they scream for) the possibility of refactoring". Machine learning is defined as computers learning to solve problems without being explicitly programmed, although they are "trained"[2]. Arthur Samuel coined the term Machine Learning in 1959[3].

Machine learning is a subset of artificial intelligence. It uses algorithms and statistical models to execute a task without explicitly being programmed. Machine learning is used in a wide variety of applications such as email spam filtering, search engines, video surveillance, and image curation.

A big challenge in this field determining the best way to identify code smells and how training datasets are labeled and created. There is a lot of inconsistency in training datasets and identifying code smells because they can be subjectively interpreted. Additionally, expert advice and knowledge may be able to help identify code smells, using it to train datasets may not be effective unless it has been experimentally shown to be result in finding code smells.

It is difficult to compare the results of previous research due to the variability of training datasets and differences in machine learning algorithms. This research will compare training datasets and code smell definitions for performance.

Figure 1 show the "potential" results of comparison by algorithm.

There are many existing public training datasets that can be compared such as the research performed by Fontana et al.[4] The different studies comparing machine learning performance with regard to code smells vary, and don't all study the same smells.

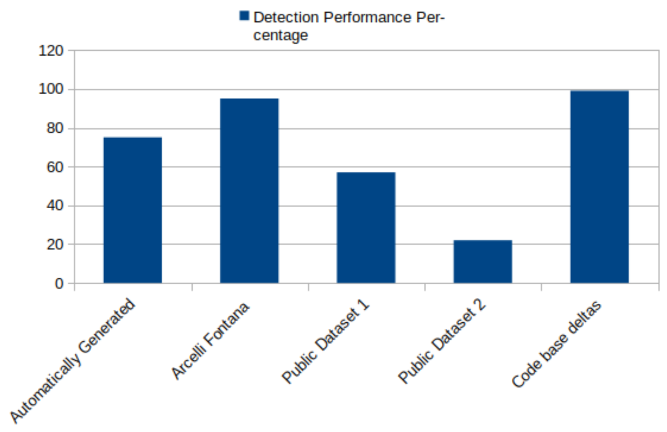


Fig. 1: Training Data Performance

II. The Approach

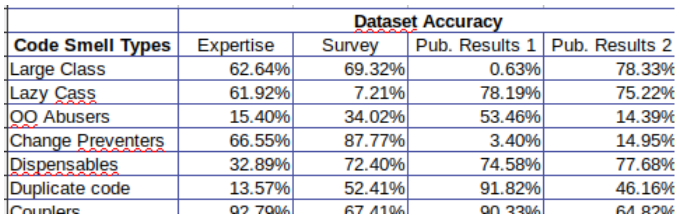
The approach to this research will be to compare training datasets for performance in detecting code smells. Common existing public datasets will be compared using the highest performing algorithms to determine which has the best performance.

III. Experimental Design

The experiment will test accuracy on performace of algorithms using different training dataset approaches. The following approaches to creating training datasets are:

1) datasets created by software experts
2) datasets created by performing a survey of software developers
3) datasets used in other published experiments
4) automated dataset generation
5) using public code base changes to identify code corrections and the implied code smells

A single machine learning algorithm will be used for all testing. From previousl research[4], the J48 and Random Forest algorithms are the highest performing when compared against other algorithms using the same training data. The experiment will compare perfomance in correctly identifying code smells and the type of smells that can be identified. An example table is shown in figure 2.



	Dataset Accuracy			
Code Smell Types	Expertise	Survey	Pub. Results 1	Pub. Results 2
Large Class	62.64%	69.32%	0.63%	78.33%
Lazy Cass	61.92%	7.21%	78.19%	75.22%
OO Abusers	15.40%	34.02%	53.46%	14.39%
Change Preventers	66.55%	87.77%	3.40%	14.95%
Dispensables	32.89%	72.40%	74.58%	77.68%
Duplicate code	13.57%	52.41%	91.82%	46.16%
Couplers	92.79%	67.41%	90.33%	64.82%

Fig. 2: Training Data Accuracy

A second part of the research will be to see if the training datasets can be optimized and improved based off of the first part of the experiment.

IV. Hypothesis and Stastical Testing

A. *Hypothesis*

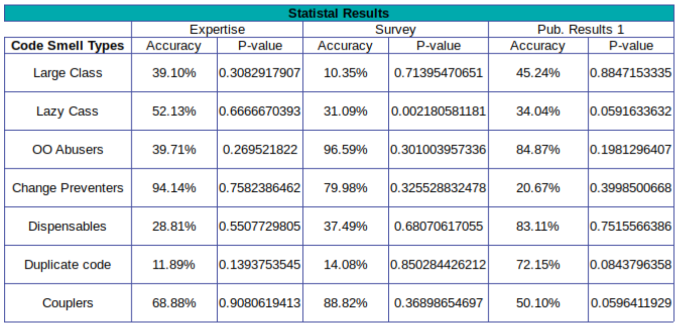
There are a few hypothesis to test:

1) Training datasets will have varying performance
2) They can more accurately identify code smells and identify a broader set of code smell types
3) Datasets can be optimized and improved

B. *Stastical Testing*

Statistical testing will be performed by comparing training datasets with regards to their behavior and coverage.

T-tests will be performed to determine if the results from the research is significant, and if so, by how much. Results will be provided, an example is provided in figure 3.

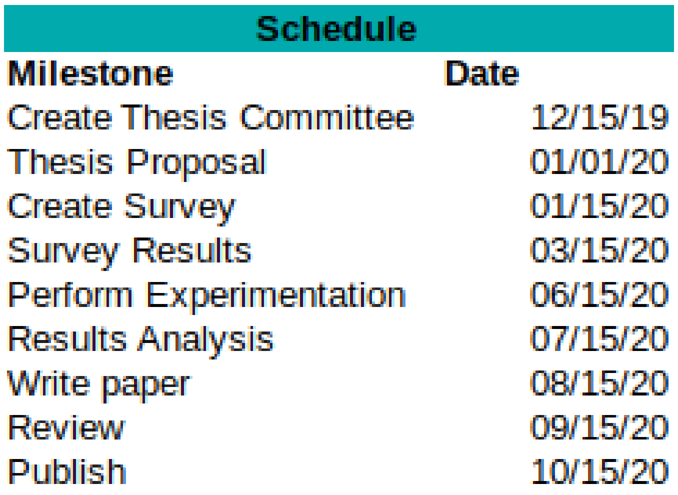


Statistal Results						
	Expertise		Survey		Pub. Results 1	
Code Smell Types	Accuracy	P-value	Accuracy	P-value	Accuracy	P-value
Large Class	39.10%	0.3082917907	10.35%	0.71395470651	45.24%	0.8847153335
Lazy Cass	52.13%	0.6666670393	31.09%	0.002180581181	34.04%	0.0591633632
OO Abusers	39.71%	0.269521822	96.59%	0.301003957336	84.87%	0.1981296407
Change Preventers	94.14%	0.7582386462	79.98%	0.325528832478	20.67%	0.3998500668
Dispensables	28.81%	0.5507729805	37.49%	0.68070617055	83.11%	0.7515566386
Duplicate code	11.89%	0.1393753545	14.08%	0.850284426212	72.15%	0.0843796358
Couplers	68.88%	0.9080619413	88.82%	0.36898654697	50.10%	0.0596411929

Fig. 3: Test Results

V. Research Plan and Timeline

Beginning Fall 2019 the project milestones are identified in figure 4.



Schedule	
Milestone	**Date**
Create Thesis Committee	12/15/19
Thesis Proposal	01/01/20
Create Survey	01/15/20
Survey Results	03/15/20
Perform Experimentation	06/15/20
Results Analysis	07/15/20
Write paper	08/15/20
Review	09/15/20
Publish	10/15/20

Fig. 4: Schedule

Scope and schedule may need to change as the research project progresses.

References

[1] M. Fowler, *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018.

[2] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[3] A. L. Samuel, "Some studies in machine learning using the game of checkers. IIrecent progress," in *Computer Games I*. Springer, 1988, pp. 366–400.

[4] F. Arcelli Fontana, M. V. Mntyl, M. Zanoni, and A. Marino, "Comparing and experimenting machine learning techniques for code smell detection," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1143–1191, Jun. 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9378-4