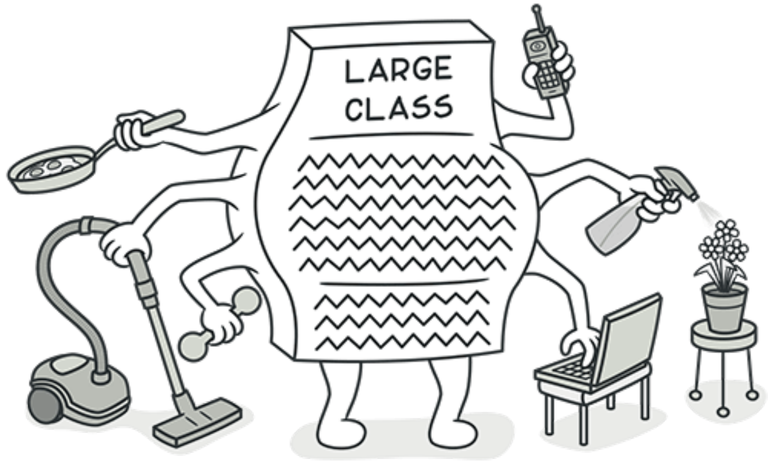
Test Smells Detection

Filipe Falcão
filipebatista@ic.ufal.br

Code Smells

- Symptoms of bad design or implementation decisions
- Not bugs and do not prevent the program from functioning
- Indicate design issues that may affect the development cycle or increase the risk of bugs (*i.e.,* a cause of technical debt)
- God Class, Feature Envy, and more [1]



1. Code Smells, 2020. Available at: https://refactoring.guru/refactoring/smells

Test Smells

- Test code is also subject to bad design/implementation choices
- Test smells may lead to bad, harder to maintain, test suites [1]
- Software tested by “smelly” suites is also more bug-prone [1, 2]

Test Smells

- Assertion Roulette
- Conditional Test Logic
- Exception Handling
- Mystery Guest
- And more [2]

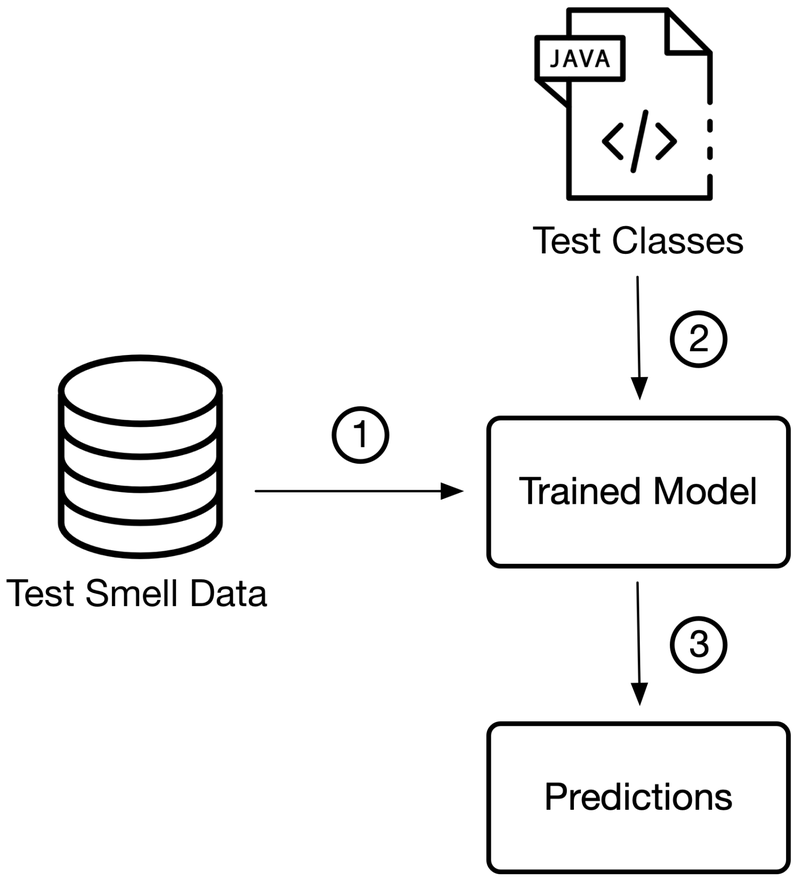
```
@Test
public void testSpinner() {
    for (Map.Entry entry : sourcesMap.entrySet()) {

        String id = entry.getKey();
        Object resultObject = resultsMap.get(id);
        if (resultObject instanceof EventsModel) {
            EventsModel result = (EventsModel) resultObject;
            if (result.testSpinner.runTest) {
                System.out.println("Testing " + id + " (testSpinner)");
                //System.out.println(result);
                AnswerObject answer = new AnswerObject(entry.getValue(), "", new CookieManager(), "");
                EventsScraper scraper = new EventsScraper(RuntimeEnvironment.application, answer);
                SpinnerAdapter spinnerAdapter = scraper.spinnerAdapter();
                assertEquals(spinnerAdapter.getCount(), result.testSpinner.data.size());
                for (int i = 0; i < spinnerAdapter.getCount(); i++) {
                    assertEquals(spinnerAdapter.getItem(i), result.testSpinner.data.get(i));
                }
            }
        }
    }
}
```

2. Software Unit Test Smells, 2020. Available at: https://testsmells.github.io/pages/testsmells.html

Test Smells Detection

- Automatically detect the presence of a test smell in a test class
- Aid testers at improving their test code
- Potentially improve the quality of the tested software

Test Smells Detection



1. Model Training (Machine Learning, Deep Learning)
2. Monitoring Test Classes
3. Predicting the occurrence of Test Smells

Experiment

- Test Smells data from *tsDetect* [1]
- Selected Test Smells: **Conditional Test Logic** & **Exception Handling**
- Data vectorization though Tokenizer [3]
- Deep Learning through a **LSTM** model [4]

3. Tokenizer, 2020. Available at: https://github.com/dspinellis/tokenizer
4. Deep Learning, 2016. Available at: https://www.deeplearningbook.org

Demo

Easy over Hard: A Case Study on Deep Learning

Wei Fu, Tim Menzies
Com.Sci., NC State, USA
wfu@ncsu.edu,tim.menzies@gmail.com

ABSTRACT

While deep learning is an exciting new technique, the benefits of this method need to be assessed with respect to its computational cost. This is particularly important for deep learning since these learners need hours (to weeks) to train the model. Such long training time limits the ability of (a) a researcher to test the stability of their conclusion via repeated runs with different random seeds; and (b) other researchers to repeat, improve, or even refute that original work.

For example, recently, deep learning was used to find which questions in the Stack Overflow programmer discussion forum can be linked together. That deep learning system took 14 hours to execute. We show here that applying a very simple optimizer called DE to fine tune SVM, it can achieve similar (and sometimes better) results. The DE approach terminated in 10 minutes; i.e. 84 times faster hours than deep learning method.

We offer these results as a cautionary tale to the software analytics community and suggest that not every new innovation should be applied without critical analysis. If researchers deploy some new and expensive process, that work should be baselined against some simpler and faster alternatives.

semantically related, they are considered as *linkable* knowledge units.

In their paper, they used a convolution neural network (CNN), a kind of deep learning method [42], to predict whether two KUs are linkable. Such CNNs are highly computationally expensive, often requiring network composed of 10 to 20 layers, hundreds of millions of weights and billions of connections between units [42]. Even with advanced hardware and algorithm parallelization, training deep learning models still requires hours to weeks. For example:

- XU report that their analysis required 14 hours of CPU.
- Le [40] used a cluster with 1,000 machines (16,000 cores) for three days to train a deep learner.

This paper debates what methods should be recommended to those wishing to repeat the analysis of XU. We focus on whether using simple and faster methods can achieve the results that are currently achievable by the state-of-art deep learning method. Specifically, we repeat XU's study using DE (differential evolution [62]), which serves as a hyper-parameter optimizer to tune XU's baseline method, which is a conventional machine learning algorithm, support vector machine (SVM). Our study asks:

***RQ1**: Can we reproduce XU's baseline results (Word Embedding + SVM)?Using such a baseline, we can compare our methods to those*

Room for Improvement

- Method-level detection
- Bigger and better dataset
- Evaluation of different neural network architectures

Thank You

References

[1] Peruma, Anthony, et al. "*tsDetect: an open source test smells detection tool.*" Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2020.

[2] Spadini, Davide, et al. "*On the relation of test smells to software code quality.*" 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2018.