Poster: Machine Learning based Code Smell Detection through WekaNose

Umberto Azadi
Università degli Studi di Milano Bicocca
Milan, Italy
u.azadi@campus.unimib.it

Francesca Arcelli Fontana
Università degli Studi di Milano Bicocca
Milan, Italy
arcelli@disco.unimib.it

Marco Zanoni
Università degli Studi di Milano Bicocca
Milan, Italy
marco.zanoni@disco.unimib.it

ABSTRACT

Code smells can be subjectively interpreted, the results provided by detectors are usually different, the agreement in the results is scarce, and a benchmark for the comparison of these results is not yet available. The main approaches used to detect code smells are based on the computation of a set of metrics. However code smell detectors often use different metrics and/or different thresholds, according to their detection rules. As result of this inconsistency the number of detected smells can increase or decrease accordingly, and this makes hard to understand when, for a specific software, a certain characteristic identifies a code smell or not. In this work, we introduce WekaNose, a tool that allows to perform an experiment to study code smell detection through machine learning techniques. The experiment's purpose is to select rules, and/or obtain trained algorithms, that can classify an instance (method or class) as affected or not by a code smell. These rules have the main advantage of being extracted through an example-based approach, rather then a heuristic-based one.

CCS CONCEPTS

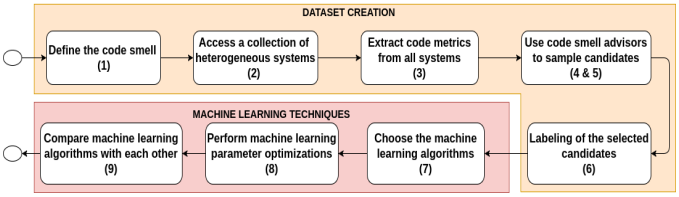
• **Software and its engineering** → **Software maintenance tools**;
• **Theory of computation** → *Machine learning theory*;

ACM Reference Format:
Umberto Azadi, Francesca Arcelli Fontana, and Marco Zanoni. 2018. Poster: Machine Learning based Code Smell Detection through WekaNose. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3194974

1 INTRODUCTION

Code smell detection is a well-known thorny issue, in fact since their identification [6] no one was able to create a definition that satisfies everyone else. The main reason is that they are quite open to interpretation and they can assume different forms based on the context that "surround" them. Usually, detection rules are based on some metrics and related thresholds, often chosen in an arbitrary way, leading to many differences in the detection results. In this work we propose a solution implemented in a tool called WekaNose that exploits supervised machine learning techniques, to support a learn-by-example process to construct code smell detection rules and/or obtain trained algorithms able to evaluate new candidates. Therefore, this tool can be used by whoever is interested into: *(i)* perform a deep study that concerns a specific code smell, *(ii)* find correlations between code smells and some specific metrics or *(iii)* create rules for rule-based code smell detectors. This experiment, synthetized in Figure 1, is divided in two main parts: *(i)* the first one concerns the creation of the dataset, discussed in Section 3 and *(ii)* the second part concerns the machine learning algorithms trained and tested using the dataset created in the first part, discussed in Section 4.

Figure 1: Flow graph of the experiment



2 RELATED WORKS

In the literature we found few works exploiting machine learning techniques for code smell detection, such as [1] [7]. The process followed by WekaNose has been already used in two studies: the first one [2] on four code smells (God Class, Long Method, Data Class and Feature Envy) and the second one [3] on two code smells (Long Parameter List and Switch Statements). In both cases it leads to the training of rule-based, decision tree and bayesian algorithms that guaranteed high performance. In those cases the process was performed "manually", which means that the result of each step described in Sections 3 and 4 was manually prepared for the next one. Hence performing the whole experiment was extremely time consuming. WekaNose allows to follow a workflow containing all the necessary steps and automatize the most tedious ones.

3 DATASET CREATION

We discuss the process that should be followed to create the dataset, i.e. a file with csv extension. The process can be separated in six steps, three of which can be performed using WekaNose.

- **Creation of a definition for the code smell** to be detected. This initial step is very important because the definition will affect the entire experiment, hence we strongly recommend to exploit some literature (e.g. [6]) to ensure the reliability of the choice.
- **Selection or creation of an heterogeneous collection of software systems**, from which the instances of the dataset can be extracted (e.g. Qualitas Corpus Repository[1]).

[1] http://qualitascorpus.com/

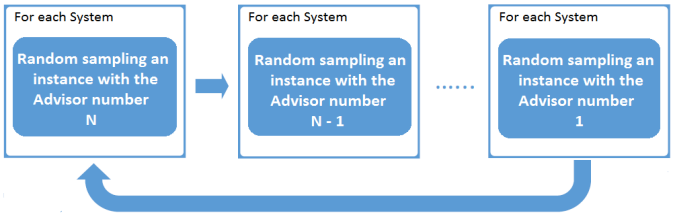
ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194974

- **Computation of a large set of object-oriented metrics** on the selected systems at class, method and package levels. This step is **performed by WekaNose** using the DFMC4J (Design Features and Metrics for Java) tool developed by our ESSeRe Lab [4], but any other tool for metrics computation can be used.
- **Choice of the Advisors through WekaNose:** an Advisor is a deterministic rule that provides a classification of a code element (method or class), telling if it is a code smell or not. The idea is that Advisors should approximate the label better than the random choice, and by aggregating their suggestions we should be able to sample more code elements affected by code smells. WekaNose implements a more generic definition which consists in a rule that should fall in one of these three main groups:
  - the ones that most likely characterise the not smell instances;
  - the ones that most likely characterise the affected instances;
  - all the other instances that cannot be easily classified.

  The rules can be extracted by some code smells detector or works in the literature, but they can affect the experiment. Hence, at least one Advisor of every group should be specified in order to balance the distribution of the instances.
- **Sampling Process through WekaNose:** the dataset elements (method or class) will be divided in groups based on the Advisors' results. After that the sampling procedure, described by Figure 2, will be done automatically by WekaNose. As result, a not labelled version of the dataset is created by WekaNose.

Figure 2: Sampling process



- **Labelling Process:** the dataset's instances have to be manually evaluated as affected or not affected by the code smell using the definitions previously created. Moreover, a set of features that should be more consistent as possible with the definitions that characterise the code smell can be gathered, for example using the literature or exploiting some available detection implementation. Then, these features can be used to support the labelling process.

4 MACHINE LEARNING TECHNIQUES

In this section the part of the experimentation that concerns the application of machine learning algorithms is discussed. We identified three steps, two of which can be performed using WekaNose. It is important to underline that WekaNose relies heavily on WEKA[2] and is designed with the aim to reduce the time needed for setting the experiment. Hence, if the experimentation requires the application of specific pre/post processing algorithms or a more customizable type of setting, more specific tools for this task can be taken in consideration, as for example WEKA itself.

- **Selection of the machine learning algorithms:** every algorithm has its strengths and its weaknesses. In our experience ([2], [3]) the paradigms that most likely lead to high performance with this type of dataset are the rule-based and decision tree.
- **Detection of the best parameters through WekaNose:** the parameters are considered the best if the performances of the machine learning algorithms are maximised by them. WEKA provides a set of algorithms[3] that perform a greed search with this purpose, that can be used in WekaNose.
- **Machine learning algorithms comparison through WekaNose:** once all the algorithms and the datasets are configured through WekaNose GUI, it is possible to "Start the experiment" [4]. This means that n-fold cross-validation tests with m repetitions will be performed for each classifier, where m and n are values that can be specified through the GUI. The result of this execution will be saved in a file that will contain specific information for each one of the $(m \cdot n)$ executions, such as Accuracy, F-measure, and Area Under ROC performance measurements. These values will be used for comparing the algorithms using the corrected paired t-tests.

5 CONCLUSION

We introduced a work-flow semi-automated through WekaNose which aims to study the code smells through a machine learning approach. The main advantage of this procedure is to select rules or trained algorithms that can classify an instance not just by a subjective way, but in a experience-based way. We underlined which steps are the most likely to affect the experimentation and we suggested to base every choice that concerns these steps on the available literature, in order to minimize their arbitrariness. In future developments we would like to consider more metrics and make WekaNose compatible with already existing code smell detectors to directly use their detection rules as Advisors. In this direction we could also think to develop a benchmark platform for comparing code smells detection results and the performances of different tools. Moreover, we could extend WekaNose by considering other code smells or other issues as antipatterns and architectural smells [5].
For any further information please visit http://essere.disco.unimib.it/wiki/wekanose and watch the Demo Tutorial[5] available online.

REFERENCES

[1] Abdou , Nasir Ali, Neelesh Bhattacharya, Aminata Sabané, Yann-Gaël Guéhéneuc, Giuliano Antoniol, and Esma Aïmeur. 2012. Support vector machines for anti-pattern detection. In *Proceedings of the 27th IEEE/ACM Inter. Conf. on Automated Software Engineering (ASE 2012)*. ACM, Essen, Germany, 278–281.
[2] Francesca Arcelli Fontana, Mika V. Mäntylä, Marco Zanoni, and Alessandro Marino. 2016. Comparing and Experimenting Machine Learning Techniques for Code Smell Detection. *Empirical Softw. Engg.* 21, 3 (June 2016), 1143–1191.
[3] Umberto Azadi. 2017. Code smell detection through machine learning techniques, Thesis, Università degli Studi di Milano-Bicocca. (July 2017).
[4] Francesca Arcelli Fontana, Vincenzo Ferme, Marco Zanoni, and Riccardo Roveda. 2015. Towards a prioritization of code debt: A code smell Intensity Index. In *7th IEEE International Workshop on Managing Technical Debt, MTD@ICSME 2015, Bremen, Germany, October 2, 2015*. 16–24. https://doi.org/10.1109/MTD.2015.7332620
[5] Francesca Arcelli Fontana, Ilaria Pigazzini, Riccardo Roveda, Damian Andrew Tamburri, Marco Zanoni, and Elisabetta Di Nitto. 2017. Arcan: A Tool for Architectural Smells Detection. In *2017 IEEE International Conf. on Software Architecture Workshops, ICSA Workshops 2017, Gothenburg, Sweden, April 5-7, 2017*. 282–285.
[6] Martin Fowler. 1999. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley Longman Publishing Co. Inc. http://www.refactoring.com/.
[7] Foutse Khomh, Stephane Vaucher, Yann-Gaël Guéhéneuc, and Houari Sahraoui. 2011. BDTEX: A GQM-based Bayesian approach for the detection of antipatterns. *The 9th International Conference on Quality Software* 84, 4 (2011), 559–572.

[2] https://www.cs.waikato.ac.nz/ml/weka/

[3] https://weka.wikispaces.com/Optimizing+parameters

[4] https://weka.wikispaces.com/Using+the+Experiment+API

[5] https://www.youtube.com/watch?v=cUKwipHZDuY