Combining Spreadsheet Smells for Improved Fault Prediction

Patrick Koch
Konstantin Schekotihin
Dietmar Jannach
AAU Klagenfurt, Austria
[firstname].[lastname]@aau.at

Birgit Hofer
Franz Wotawa
Graz University of Technology
Austria
{bhofer,wotawa}@ist.tugraz.at

Thomas Schmitz
TU Dortmund
Germany
thomas.schmitz@tu-dortmund.de

ABSTRACT

Spreadsheets are commonly used in organizations as a programming tool for business-related calculations and decision making. Since faults in spreadsheets can have severe business impacts, a number of approaches from general software engineering have been applied to spreadsheets in recent years, among them the concept of code smells. Smells can in particular be used for the task of fault prediction. An analysis of existing spreadsheet smells, however, revealed that the predictive power of individual smells can be limited. In this work we therefore propose a machine learning based approach which combines the predictions of individual smells by using an AdaBoost ensemble classifier. Experiments on two public datasets containing real-world spreadsheet faults show significant improvements in terms of fault prediction accuracy.

KEYWORDS

Spreadsheet Smells, Spreadsheet QA, Fault Prediction

ACM Reference Format:
Patrick Koch, Konstantin Schekotihin, Dietmar Jannach, Birgit Hofer, Franz Wotawa, and Thomas Schmitz. 2018. Combining Spreadsheet Smells for Improved Fault Prediction. In *Proceedings of 40th International Conference on Software Engineering: New Ideas and Emerging Results Track, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE-NIER'18),* 4 pages.
https://doi.org/10.1145/3183399.3183402

1 INTRODUCTION

Many decisions in organizations are based on spreadsheets. One reason for the broad success of spreadsheets is their simple and intuitive computation paradigm, which allows even end users to develop spreadsheet programs according to their needs. However, these programs are particularly prone to faults for two main reasons: *(i)* most of the users have no or only little background in general software development, and *(ii)* today's spreadsheet environments have limited support for quality assurance (QA). The resulting faults can lead to substantial financial losses for companies.[1]

Various quality assurance approaches for spreadsheets were suggested in recent years, including techniques for visualization, testing, debugging, and fault prevention [12]. *Spreadsheet smells* are a prominent approach that can be particularly helpful in the context of fault prevention, e.g., in preventive maintenance or fault prediction. They transfer the idea of *code smells* [6] to the spreadsheet domain and represent heuristics that are designed to indicate potential problems in spreadsheets such as complex formulas, possibly missing inputs, and problematic dependencies [1, 3, 10, 11].

Abreu *et al.* [1] relied on a combination of spreadsheet smells and other techniques for *fault prediction*. In particular, they used smells to derive a fault likelihood for each cell in a spreadsheet. Our work continues this general line of research on fault prediction using smells. While Abreu *et al.* considered individual smells as equal in terms of predictive power, our research indicates that *(i)* the fault prediction power varies significantly across different smells, and *(ii)* the predictive power of individual smells is comparably low.

In this work, we therefore propose a novel smell-based fault prediction approach for spreadsheets that is based on learning optimal combinations of smells with machine learning (ML) techniques. Technically, we frame the smell-based fault prediction problem as a supervised classification problem. The inputs to the ML problem are *(i)* a set of spreadsheets as training data for which the faulty formulas are known and *(ii)* a set of smells from the literature as fault predictors. The overall process of making predictions then consists of the following main steps. First, we compute the "strength" of each given smell for all formulas that are contained in the training spreadsheets. These smell values, together with a label (correct or faulty), for each formula are then used as training data for the learning problem. Given that form of data representation, a variety of supervised ML algorithms can be applied to learn a function to predict the fault probabilities of unlabelled formulas.

We tested our method on two publicly available datasets of real-world spreadsheets for which the faulty formulas are known. The experimental evaluation showed that an ensemble method, AdaBoost [7], led to the best classification results and outperformed fault predictors that were based on individual smells by far. The obtained absolute *recall* values ranged between 70 % and 95 %, which indicates that a large majority of the existing faults can be identified by the smell-based ensemble predictor.

2 RELATED WORK

The work of Abreu *et al.* [1] is the contribution that is most closely related to ours. As mentioned in the introduction, the authors use smells as part of their fault prediction approach for spreadsheets. Similar to our work, they rely on a set of smells from the literature and, in a first step, compute the strength of each smell for each cell in the given spreadsheet(s). The subsequent steps are, however, different from our work. Abreu *et al.* apply a threshold for each computed measure that classifies each cell as being smelly or not.

[1] See http://www.eusprig.org/horror-stories.htm for a list of examples.

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

ACM ISBN 978-1-4503-5662-6/18/05.
https://doi.org/10.1145/3183399.3183402

They then compute the set of output cells (cells that are not referred to), as well as the calculation chains of these cells. In a final step, Spectrum-based Fault Localization (SFL) is used to compute the suspiciousness of each cell. Cells that are often involved in calculation chains of smelly cells and less often in calculation chains of non-smelly cells are more suspicious of being faulty.

Singh *et al.* [18] proposed an approach to use machine learning methods for fault prediction for spreadsheets. In their tool, named Melford, a neural network is trained with a set of custom engineered features, based on the structure and content of spreadsheets, in order to predict "number-where-formula-expected" faults. Differently from their work, our approach *(i)* uses spreadsheet smells from the literature as features, *(ii)* applies a different learning model, and *(iii)* is not limited to certain types of faults.

A number of previous works considered code smells as part of the fault prediction process in the general field of software engineering. Fontana *et al.* [5], for example, applied ML algorithms to detect code smells in software systems. Palomba *et al.* [16] improved the performance of a bug prediction system based on smells by introducing the concept of "smell intensity levels". Ma *et al.* [14] used fault prediction based on smells to guide the refactoring of code. While these approaches aim to improve prevalent software QA practices, they were not designed to consider the specific types of potential problems that can be found in spreadsheet programs.

3 TECHNICAL APPROACH

In this section, we provide the technical details of how we framed the fault prediction problem as a *supervised classification* problem using spreadsheet smells, how we preprocessed the data, and how we optimized the used prediction models.

Problem Definition and Data Preprocessing. The fault prediction problem can be summarized as follows: Given *(i)* a set of faulty spreadsheets in which every formula is labeled either as being faulty or correct, and *(ii)* a number of smells, learn a function that predicts whether some previously unseen formula is faulty or not.

Supervised ML techniques use a set of training examples, where each example is characterized by a set of features and has one label assigned. In our case, each example is constructed for a formula of a training spreadsheet and its label indicates the formula being faulty or correct. The set of features corresponds to the set of smells that are used in the learning problem. The feature values (called the feature vector) for each example are determined by computing the strength of each smell for the related formula. Table 1 shows the general structure of the problem encoding.

To build the table of training data, for each formula we compute the strength of each smell according to the heuristics from the literature, and assign the appropriate label provided in the input. The resulting training data table is complete, i.e., no value is missing.

Model Optimization & Learning. Given these inputs, a variety of machine learning approaches can be applied, optimizing some given performance measure. Since the given classification problem is binary (faulty or correct), we optimize our models for the F1-measure, which is a standard classification accuracy measure that is computed as the harmonic mean of *precision* and *recall*.

Table 1: Structure of the training data

Cell	$Smell_1$	...	$Smell_n$	*Label*
$cell_1$	$value_{1,1}$	...	$value_{n,1}$	correct/faulty
...	...	...	...	...
$cell_n$	$value_{1,n}$	...	$value_{n,n}$	correct/faulty

We tested various ML techniques for the given problem. The best results in terms of F1-measure and high recall were achieved when we used *Adaptive Boosting* (AdaBoost) [7], and we therefore use it as representative in our evaluation. AdaBoost is a meta-algorithm that combines the output of many, possibly individually weak, classifiers (in our case decision trees) to obtain a better classification outcome.

Supervised learning techniques allow for the fine-tuning of an optimization goal for the given data using model-specific parameters. In our experiments, we apply a grid search method to explore all possible values from a given set and pick the one that leads to the highest value of the F1-measure. In the case of AdaBoost, the main parameter to be set was the number of used decision trees.

We use 10-fold cross-validation for optimizing and evaluating our models. To avoid that results are dependent on the choice of the partitioning into training and test examples, we apply stratified folding with shuffling, guaranteeing a mixed but roughly equal distribution of samples of both classes within each fold. Before processing, the feature values are standardized, shifting the data for each feature to zero mean and scaling it to Gaussian unit variance in order to meet the requirements of the used supervised learning approaches. Finally, since the number of correct formulas in the input spreadsheets is significantly higher than the number of faulty ones, models trained with this data might be biased to predict an input formula to be correct. Therefore, we use an *oversampling* procedure which is applied to the training data of every fold. Specifically, we generate additional training examples from the minority class, i.e., cases that are labeled as faulty, by adding copies of randomly picked faulty examples until the number of examples in each class is equal.

4 EVALUATION

We performed experiments on two datasets to assess the effectiveness of our smell-based fault prediction method. We recorded the F1-measure using a 10-fold cross-validation procedure, and compare our performance results using AdaBoost with those that were achieved when individual smells were used either as predictors or as part of a voting committee, and with the results of using an alternative machine learning method. To enable validation and replicability of our research, we share the source code used in the experiments and the detailed results for all datasets online[2].

4.1 Study Setup

Datasets. The first dataset is based on a subset of the Enron spreadsheet corpus [9] which contains real-world faults [17]; the detailed list of faults can be found online.[3] Overall, the Enron Errors Corpus contains 26 spreadsheets with faulty formulas, with 2.9 % of the formulas – 481 out of 16,790 – being faulty. The second dataset (called "INFO1") [8] contains spreadsheets developed by civil engineering

[2] http://spreadsheets.ist.tugraz.at/wp-content/uploads/2018/01/ICSE18.zip
[3] http://ls13-www.cs.tu-dortmund.de/homepage/spreadsheets/enron-errors.htm

students as part of an exercise. It comprises 119 spreadsheets with 5,157 faulty formulas (3.0 %). More details can be found online.[4]

Used Smells. We used a set of 19 spreadsheet smells and the corresponding strength calculation rules that were proposed in previous research [1, 3, 10, 11]. In general, the strength of a specific smell is expressed by the value of a related complexity metric that is measured for the given formula or worksheet. The detailed list of smells used in the study is shown in Table 2. Since we focus on the prediction of faulty formulas, we included formula and worksheet smells and did not consider smells for data cells. The measurements of the worksheet smells were applied to each formula of the worksheet.

Table 2: Overview of Used Smells

Index	Name	Target
0	Column-wise Pattern Finder [3]	cell
1	Row-wise Pattern Finder [3]	cell
2	Reference to empty cells [3]	cell
3	Changing Formulas [10]	cell
4	Changing Worksheets [10]	cell
5	Duplicated Calculations [1]	cell
6	Duplicated Formulas [11]	cell
7	Feature Envy [10]	cell
8	Long Calculation Chain [11]	cell
9	Conditional Complexity [11]	cell
10	Multiple Operations [11]	cell
11	Multiple References [11]	cell
12	Inappropriate Intimacy [10]	worksheet
13	Middle Man [10]	worksheet
14	Shotgun Surgery (Formulas) [10]	worksheet
15	Shotgun Surgery (Worksheets) [10]	worksheet
16	Inconsistent Formula Group Reference[13]	worksheet
17	Missing Header[13]	worksheet
18	Overburdened Worksheet[13]	worksheet

Baseline Methods. To assess the performance of the AdaBoost cla sifier, we compare it with three types of baselines. The first typ uses individual smells and implements a simple classification rul Given a formula, a spreadsheet smell, and a threshold percentag T, it classifies the formula as faulty if the computed strength o the smell lies above the lowest T % of all feature values for the smell. We determined the optimal value for T for each of the 19 smells through a grid search method. Following the suggestion of Hermans *et al.* [10], we tested three threshold values for T (70 %, 80 %, and 90 %). For the second baseline, we combined the optimized predictions of individual smells using two simple voting schemes: *(i)* majority voting using uniform weights (called "Voting: majority"), and *(ii)* advocate voting, where any smell classifier voting for faulty suffices for the ensemble to classify a sample as faulty (called "Voting: advocate"). As the third baseline, we use linear Support Vector Machines (SVMs), as they were found to be effective for comparable learning tasks [5, 15]. To deal with the computational complexity in particular for the larger INFO1 dataset, we chose Stochastic Gradient Descent (SGD) as the learning method for the used linear SVMs, which is recommended for large-scale training of classifiers [2]. We optimized the regularization parameter α of the SGD training process through a systematic grid search. The first two baselines model a scenario in which a user manually selects one smell or a combination of smells for fault detection. The third baseline offers a comparison with another established ML approach.

Parameter Selection. For the AdaBoost classifier, the grid search using a 10-fold cross-validation procedure returned 5 as the optimal number of decision trees for the Enron Errors Corpus and 1 for the INFO1 corpus. The optimized T values for the individual smell classifiers can be inspected in the analysis script provided online. The "voting" classifiers use the already optimized classifiers of individual smells. For the SVM baseline with SGD, using a regulatory parameter of 0.0001 led to the best results for the Enron dataset; 0.001 was the optimal setting for the INFO1 dataset.

4.2 Results and Discussion

Figure 1 shows the results for precision (x-axis), recall (y-axis) and the F1-measure (radial line) obtained for the Enron Errors Corpus. Smells that target cells are represented by triangles, worksheet-based smells are represented as squares, and the results of voting and ensemble classifiers are indicated by 'x' symbols.

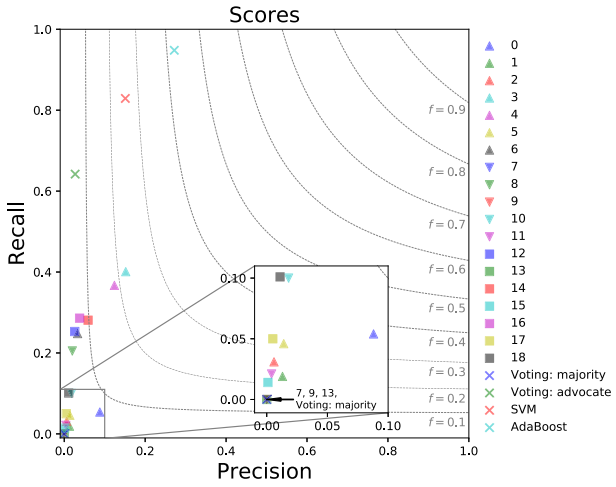

Figure 1: Precision-recall performance for the Enron Errors Corpus. The numbers in the legend correspond to the indices given in Table 2.

The proposed ensemble learning approach, AdaBoost, significantly outperforms the baseline techniques. The obtained recall value is at about 95 %, which means that the majority of faults was successfully identified by our method. The precision of about 30 % implies that two out of three fault predictions are "false alarms". Whether precision or recall is more important depends on the domain or application scenario. In our case, the main goal is not to miss faults that otherwise would remain in the spreadsheets. Hence, high recall values are particularly desirable. While a "false alarm" rate of 70 % might seem a lot, consider that each of the examined spreadsheets contains usually only one to three faults. Hence, only about 3 to 9 of possibly hundreds of formulas have to be inspected.

[4] http://spreadsheets.ist.tugraz.at/index.php/corpora-for-benchmarking/info1/

Moreover, these results were achieved using a fixed set of smell-based predictors and a limited set of training data. Diversification and optimization of the used features, as well as the use of additional training data might further improve the precision scores.

While outperformed by AdaBoost, SVM performs well, achieving a recall of about 83 %, and a precision of about 15 %. This confirms the conjecture that more elaborate ensemble methods generally perform better than any single classifier [4].

In comparison, the simple combination of smell classifiers by means of voting schemes, as indicated by the results of the "voting" classifiers, lead to poor prediction performance. Majority voting did not detect any faults, as no majority was found for any of the faulty cells. Advocate voting achieved a recall of about 65 %, but only a precision of about 3 %. This reveals the major shortcomings of simple ensemble schemes using smells: no single threshold-based smell classifier is capable of detecting all faults, and no fault case is pronounced enough for a majority of smells to indicate it.

Many of the individual smells have limited predictive power when used in isolation, leading to low recall and precision values. Smells that are measured per formula cell, barring some exceptions, generally exhibit limited prediction performance. Smells that are measured per worksheet slightly outperform the majority of per-formula smells in terms of recall, but also lack precision.

Overall, the use of isolated smells and their simple combinations is not very helpful for fault prediction, whereas combining them as proposed in this work leads to substantially higher predictive power. This indicates that actual faults in spreadsheets emerge from a combination of specific deficiencies which are difficult to capture by means of simple metric thresholds.

The evaluation on the INFO1 dataset led to comparable results[5]: The best performing classifier is AdaBoost (recall: 71 %, precision: 30 %, F1: 0.42). While SVM and the advocate voting ensemble have a higher recall (77 % respectively 78 %), their precision is significantly lower (7 % and 3 %), as is their F1 score (0.12 and 0.06). The majority voting ensemble has both a precision and a recall of 0 %. The F1-measure of all individual smell classifiers is below 0.1. These results confirm the ones we have obtained for the Enron Errors Corpus.

4.3 Threats to Validity

The main threat to the *internal* validity of our research is related to the correctness of the software used for analysis and evaluation. To allow other researchers to validate our work, all source code and the used datasets are provided online. The main threat to the *external* validity of our study is the representativeness of the used spreadsheet corpora with regard to the overall population of faulty spreadsheets. Generally, the Enron spreadsheets used in the study have been used extensively for empirical research in previous works. The specific set of real-world faults in the corpus was furthermore obtained in a systematic and reproducible manner [17] and we therefore consider the risk that the faults are not representative as low. The representativeness of the INFO1 corpus might be limited as it contains spreadsheets that were designed for the same problem specification. Nonetheless, since the obtained results are very similar for both datasets, we are confident that the observations obtained with this dataset are reliable as well.

[5] A precision-recall plot can be found in the online material.

5 CONCLUSIONS & OUTLOOK

Our work shows that spreadsheet smells can be valuable instruments for fault prediction in spreadsheets when they are not considered in isolation. In general, we consider the application and further development of modern and powerful machine learning methods for spreadsheet quality assurance as an emerging and promising area, in particular as past approaches to spreadsheet QA were often based on heuristics for fault identification and repair that were designed based on domain expertise.

From an algorithmic perspective, our next steps include the investigation of alternative learning models, in particular deep-learning techniques, the application of feature selection methods to identify and remove noisy smells, and the exploration of alternative methods for oversampling. Regarding the general approach, we plan to investigate the performance of additional types of smells and other spreadsheet quality metrics as fault predictors.

ACKNOWLEDGMENTS

The work described in this paper has been been funded by the Austrian Science Fund (FWF) project *DEbugging Of Spreadsheet programs (DEOS)* under contract number I2144 and the Deutsche Forschungsgemeinschaft (DFG) under contract number JA 2095/4-1.

REFERENCES

[1] Rui Abreu, Jácome Cunha, João Paulo Fernandes, Pedro Martins, Alexandre Perez, and João Saraiva. 2014. Smelling Faults in Spreadsheets. In *Proc. ICSME '14*. 111–120.
[2] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proc. COMPSTAT '10*. 177–186.
[3] Jácome Cunha, João Paulo Fernandes, Hugo Ribeiro, and João Saraiva. 2012. Towards a Catalog of Spreadsheet Smells. In *Proc. ICCSA '12*. 202–216.
[4] Thomas G Dietterich. 2000. Ensemble methods in machine learning. *Multiple Classifier Systems* 1857 (2000), 1–15.
[5] Francesca Arcelli Fontana, Mika V Mäntylä, Marco Zanoni, and Alessandro Marino. 2016. Comparing and experimenting machine learning techniques for code smell detection. *Empirical Software Engineering* 21, 3 (2016), 1143–1191.
[6] Martin Fowler. 1999. *Refactoring - Improving the Design of Existing Code.* Addison-Wesley.
[7] Yoav Freund, Robert Schapire, and Naoki Abe. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence* 14, 771-780 (1999).
[8] Elisabeth Getzner, Birgit Hofer, and Franz Wotawa. 2017. Improving Spectrum-Based Fault Localization for Spreadsheet Debugging. In *Proc. QRS '17*. 102–113.
[9] Felienne Hermans and Emerson R. Murphy-Hill. 2015. Enron's Spreadsheets and Related Emails: A Dataset and Analysis. In *Proc. ICSE '15*. 7–16.
[10] Felienne Hermans, Martin Pinzger, and Arie van Deursen. 2012. Detecting and visualizing inter-worksheet smells in spreadsheets. In *Proc. ICSE '12*. 441–451.
[11] Felienne Hermans, Martin Pinzger, and Arie van Deursen. 2012. Detecting code smells in spreadsheet formulas. In *Proc. ICSM '12*. 409–418.
[12] Dietmar Jannach, Thomas Schmitz, Birgit Hofer, and Franz Wotawa. 2014. Avoiding, finding and fixing spreadsheet errors - A survey of automated approaches for spreadsheet QA. *Journal of Systems and Software* 94 (2014), 129–150.
[13] Patrick W. Koch. 2016. *Smelly Spreadsheet Structures: Structural Analysis of Spreadsheets to enhance Smell Detection.* Master's thesis. Graz University of Technology, Austria. http://spreadsheets.ist.tugraz.at/wp-content/uploads/2016/05/DA_thesis_final.pdf
[14] Wanwangying Ma, Lin Chen, Yuming Zhou, and Baowen Xu. 2016. Do We Have a Chance to Fix Bugs When Refactoring Code Smells?. In *Proc. SATE '16*. 24–29.
[15] A. Maiga, N. Ali, N. Bhattacharya, A. Sabané, Y. G. Guéhéneuc, G. Antoniol, and E. Aïmeur. 2012. Support vector machines for anti-pattern detection. In *Proc. ASE '12*. 278–281.
[16] Fabio Palomba, Marco Zanoni, Francesca Arcelli Fontana, Andrea De Lucia, and Rocco Oliveto. 2016. Smells like teen spirit: Improving bug prediction performance using the intensity of code smells. In *Proc. ICSME '16*. 244–255.
[17] Thomas Schmitz and Dietmar Jannach. 2016. Finding Errors in the Enron Spreadsheet Corpus. In *Proc. VL/HCC '16*. 157–161.
[18] Rishabh Singh, Benjamin Livshits, and Benjamin Zorn. 2017. *Melford: Using Neural Networks to Find Spreadsheet Errors.* Technical Report MSR-TR-2017-5. Microsoft.