Handling Uncertainty in Code Smells Detection using a Possibilistic SBSE Approach

Sofien Boutaib, Slim Bechikh
SMART Lab, University of Tunis, ISG, Tunis, Tunisia
boutaibsofien@yahoo.fr
slim.bechikh@fsegn.rnu.tn

Carlos A. Coello Coello
CINVESTAV-IPN, Mexico City, Mexico
ccoello@cs.cinvestav.mx

Chih-Cheng Hung
Kennesaw State University, Marietta, GA, USA
chung1@kennesaw.edu

Lamjed Ben Said
SMART Lab, University of Tunis, ISG, Tunis, Tunisia
lamjed.bensaid@isg.rnu.tn

ABSTRACT

Code smells, also known as anti-patterns, are indicators of bad design solutions. However, two different experts may have different opinions not only about the smelliness of a particular software class but also about the smell type. This causes an uncertainty problem that should be taken into account. Unfortunately, existing works reject uncertain data that correspond to software classes with doubtful labels. Uncertain data rejection could cause a significant loss of information that could considerably degrade the performance of the detection process. Motivated by this observation and the good performance of the possibilistic K-NN classifier in handling uncertain data, we propose in this paper a new evolutionary detection method, named ADIPOK (Anti-pattern Detection and Identification using Possibilistic Optimized K-NN), that is able to cope with the uncertainty factor using the possibility theory. The comparative experimental results reveal the merits of our proposal with respect to four relevant state-of-the-art approaches.

CCS CONCEPTS

• **Software and its engineering** → **Search-based software engineering**; **Maintaining software**;

KEYWORDS

Code smells detection, uncertain class labels, possibilistic K-NN, evolutionary algorithm

ACM Reference Format:
Sofien Boutaib, Slim Bechikh, Carlos A. Coello Coello, Chih-Cheng Hung, and Lamjed Ben Said. 2020. Handling Uncertainty in Code Smells Detection using a Possibilistic SBSE Approach. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3389948

1 INTRODUCTION

Despite the high number of works on code smells (anti-patterns) detection in the SE (Software Engineering) literature, this problem is still so far a challenging and timely research topic [9]. Many detection methods have been proposed in the literature [9] such as rule-based, machine learning-based, and search-based ones. The latter has shown its superiority over the others mainly thanks to its global search ability. Similarly to the data classification problem, the code smells detection could have uncertain class labels. Indeed, software engineers could have contradicting opinions about the smelliness of a particular software class or not. Moreover, the same observation applies for the anti-pattern type identification. Unfortunately, almost all existing works, including SBSE (Search-Based Software Engineering) ones, do not consider the uncertainty issue. Rejecting uncertain data may incur a loss of information [8] and significantly downgrades the detection performance. Motivated by the interesting performance of the PK-NN (Possibilistic K-Nearest Neighbor) algorithm [8] in classifying uncertain data, we propose in this paper a new search-based method, named ADIPOK, which evolves a population of smells detectors each corresponding to a PK-NN classifier using a GA (Genetic Algorithm).

2 POSSIBILITY THEORY

Possibility theory is a popular uncertainty theory used to represent imperfect information. Given the universe of discourse $\Omega=\{\omega_1, \omega_2,..., \omega_n\}$, the possibility distribution, denoted by π, is one of the basic concepts in possibility theory. It attributes to every state ω_i belonging to the universe of discourse Ω, a value from the possibilistic scale L to obtain the so-called possibility degree. The latter is considered as an encoding of our knowledge on the real world. Note that the possibilistic scale L is usually defined within the unit interval [0, 1], where the possibility distribution values make sense. By convention, for whatever state, $\omega_i \in \Omega, \pi(\omega_i) = 0$ signifies that the realization of ω_i is impossible, while $\pi(\omega_i) = 1$ signifies that achieving ω_i is totally possible in the real world. Furthermore, it is assumed that π is normalized when there is at least one state ω_i, where $\pi(\omega_i) = 1$ (i.e., totally possible). Over this paper, only normalized possibility distributions are adopted. In possibility theory, there are two extreme forms of knowledge that are: (a) the complete knowledge ($\exists\ \omega_k$, $\pi(\omega_k) = 1$ and, $\pi(\omega) = 0, \forall\ \omega \neq \omega_k$), and (b) the total ignorance ($\forall\ \omega \in \Omega$, $\pi(\omega) = 1$)).

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7127-8/20/07…$15.00
https://doi.org/10.1145/3377929.3389948

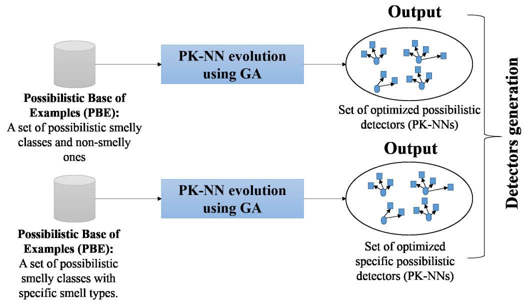
Output
Possibilistic Base of Examples (PBE):
A set of possibilistic smelly classes and non-smelly ones
PK-NN evolution using GA
Set of optimized possibilistic detectors (PK-NNs)
Output
Possibilistic Base of Examples (PBE):
A set of possibilistic smelly classes with specific smell types.
PK-NN evolution using GA
Set of optimized specific possibilistic detectors (PK-NNs)
Detectors generation

Figure 1: The main scheme of the ADIPOK approach.

3 PROPOSED APPROACH: ADIPOK

The first step to develop our ADIPOK approach is the construction of the possibilistic Bases of Examples (BEs). In this study, we considered six open-source software projects (GanttProject, ArgoUML, XercesJ, JFreeChart, AntApache, Azureus), 23 quality metrics [5] (which are widely used in the literature), and 8 smell types (God Class, Data Class, Feature Envy, Long Method, Long Parameters List, Spaghetti Code, Functional Decomposition, Duplicated Code) [5]. The Naïve Bayes classifier is employed to predict the probability distribution for each instance (i.e., software class) and then a mathematical conversion is applied using an equation (proposed in [1]) to convert the obtained Probability distribution into a Possibility one. As illustrated by Figure 1, once the BEs (i.e., datasets) are generated, the GA evolves a set of PK-NN classifiers by optimizing a fitness function (i.e., the Possibilistic F-measure (PF-measure)) that is computed based on the BEs. Once the set of optimized possibilistic detectors are generated, they are used to detect and/or identify the existing smells on unseen software systems based on the AFO (Adaptive Fusion Operator) [2] to aggregate the different possibility distributions. In fact, as our approach is based on the GA to evolve a population of smell detectors (PK-NN), the GA operators should be presented such as: {the solution representation, the fitness function, the selection mechanism, and the variation operators}. The *Solution representation*, is a vector containing four parameters: the FS (Feature Subset) is a binary vector, the K (number of nearest neighbors) is an integer in $[1, T]$, λ and α are real numbers in $[0, 1]$. As for the *Fitness Function*, code smells detection usually corresponds to an imbalanced binary classification problem. Based on this fact, we have inspired our metric from F-measure classification metric (as it is not suitable for the case of uncertain class labels) to propose a possibilistic version of this metric named PF-measure and expressed by equation (1). The π_{init} is the initial possibility distribution of the test set, π_{result} is the possibility distribution produced by the PK-NN also on the test set, and $\bigtriangleup$ is an operator that performs the intersection between π_{init} and π_{result}. As for the *selection mechanism*, we adopted the binary tournament selection operator. As for the *variation operators*, the crossover is performed per part on the chromosome: using the SBX operator (for the first and last part of the chromosome) and the uniform crossover for the remaining part. The mutation operation is also applied per part using one point mutation for the first part of the chromosome and the polynomial mutation for the remaining parts.

$$PF-measure = \pi_{result}(C_j) \ \bigtriangleup \ \pi_{init}(C_j) \tag{1}$$

4 EXPERIMENTAL VALIDATION

We compared our ADIPOK approach to the relevant state-of-the-art approaches such as: DECOR [4], GP [6], and BLOP [7]. Moreover, we evaluated our approach based on a five-fold cross-validation procedure. The parameter settings of ADIPOK approach were tuned as follows: the population size is equal to 200, the crossover and mutation rates are equal to 0.9, 0.1 respectively. In addition to the PF-measure, we used the *IAC* (Information Affinity-based Criterion) [3] to quantify the performance of the different considered methods as it takes into account non-plausible class labels membership degrees, which is not the case of the PF-measure. The IAC metric quantifies the distance between the initial possibility distribution π_{init} and the resulting π_{result}. For the detection and identification processes, our approach outperforms all the remaining approaches on all the used software projects as the compared approaches are not able to manage the uncertain class labels or to handle the data imbalance issues. The outperformance of ADIPOK could be explained as follows. For the uncertain case, the PF-measure is a good choice to deal with uncertain class labels. For the certain case, the PF-measure is the same as the F-measure, which is a suitable metric for data imbalance.

5 CONCLUSION

In this paper, we have tackled an important topic that is usually neglected in the SBSE community, which corresponds to the uncertainty of software class labels in code smells detection To do so, we have proposed ADIPOK as a new method and tool that is able to effectively detect and identify anti-patterns in both uncertain and crisp environments. As Future work, we propose to manage the uncertainty that could be present within the feature level in addition to the class label level. It would be very important to extended ADIPOK to deal with such a challenging situation.

REFERENCES

[1] Didier Dubois and Henri Prade. 1985. Unfair coins and necessity measures: towards a possibilistic interpretation of histograms. *Fuzzy sets and systems* 10, 1-3 (1985), 15–20.

[2] Didier Dubois and Henri Prade. 1994. La fusion d'informations imprécises. *TS. Traitement du signal* 11, 6 (1994), 447–458.

[3] Ilyes Jenhani, Nahla Ben Amor, Zied Elouedi, Salem Benferhat, and Khaled Mellouli. 2007. Information affinity: A new similarity measure for possibilistic uncertain information. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, Vol. 4724. Springer, 840–852.

[4] Naouel Moha, Yann-Gaël Guéhéneuc, Laurence Duchien, and Anne-Francoise Le Meur. 2009. Decor: A method for the specification and detection of code and design smells. *IEEE Transactions on Software Engineering* 36, 1 (2009), 20–36.

[5] Ali Ouni. 2014. *A Mono-and Multi-objective Approach for Recommending Software Refactoring*. Dissertation. Department of Computing Science, Faculty of Montreal.

[6] Ali Ouni, Marouane Kessentini, Houari Sahraoui, and Mounir Boukadoum. 2013. Maintainability defects detection and correction: a multi-objective approach. *Automated Software Engineering* 20, 1 (2013), 47–79.

[7] Dilan Sahin, Marouane Kessentini, Slim Bechikh, and Kalyanmoy Deb. 2014. Code-smell detection as a bilevel problem. *ACM Transactions on Software Engineering and Methodology* 24, 1 (2014), 1–44.

[8] Sarra Saied and Zied Elouedi. 2018. K-Nearest Neighbors Under Possibility Framework with Optimizing Parameters. In *Proceedings of the International Conference on Intelligent Systems Design and Applications*. Springer, 354–364.

[9] Tushar Sharma and Diomidis Spinellis. 2018. A survey on software smells. *Journal of Systems and Software* 138 (2018), 158–173.