Software defect prediction based on features extracted from the tree structure of the source code and from its development process

Lucija Šikić*, M.Eng.
Advisor: Associate professor Marin Šilić, Ph.D.
University of Zagreb Faculty of Electrical Engineering and Computing

1. Introduction

The development of software defect prediction (SDP) models is of great importance for improving software quality. Such models are used to identify defect-prone software files represented by a set of features so that these modules can be corrected prior to the testing process, which ultimately helps to optimize the allocation of testing resources.

2. Problem Description

SDP models learn from previous project version files to identify defect-prone current project version files, where the previous and current project version files are represented by features of the same type. However, most of the existing models are not specialized for the task of defect prediction nor for processing the source code representations of files. Besides, the existing metrics used to represent files do not reflect the entire file development process, and such information may be relevant to defect-proneness.

3. Methodology

To address the gaps in existing approaches, we propose a new set of metrics [Šikić *et al.*, 2021] and an end-to-end model specifically designed for processing abstract syntax trees (ASTs) representing the source code of files [Šikić *et al.*, 2022].

A new set of fourteen metrics, called aggregated change metrics (ACMs), is computed by aggregating change metrics that describe commits that change files between the analyzed versions, taking into account the chronological order of those commits, as shown in Figure 1.

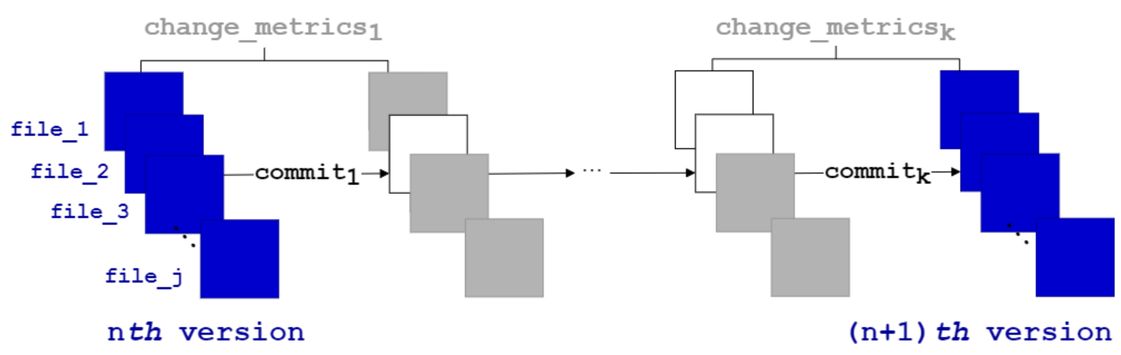

Figure 1. Collecting data for computing ACMs.

The proposed model, named DP-GCNN, is a spectral-based graph convolutional neural network that learns to perform a binary classification of files represented by a graphical representation of ASTs. DP-GCNN adapts its architecture, which is shown in Figure 2, to the analyzed project so that it can capture the same amount of information regardless of the size of the project modules.

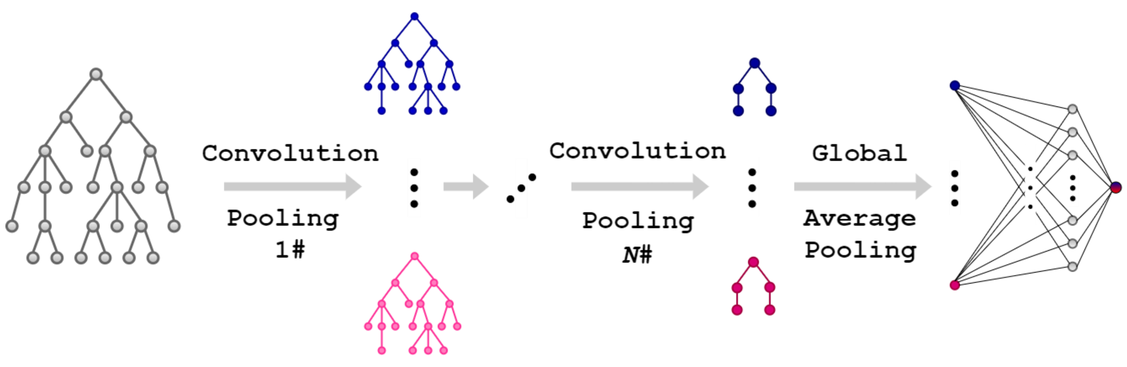

Figure 2. Classifying an AST using DP-GCNN.

4. Results

Files from two consecutive versions of seven open-source Java projects are used to build and then evaluate a random forest classifier (RFC) with the embedded feature selection method and DP-GCNN. For each file from both project versions, we collected the corresponding source code and a vector of 20 object-oriented metrics (OOMs) describing the file. To represent the files in a form suitable for DP-GCNN, we first translated the source code of each file into AST and then into the corresponding adjacency matrix and node feature matrix.

For each project, RFC selected 20 features with the highest Gini importance from a set of OOMs and ACMs. The histogram in Figure 3 shows that some of the ACMs can give the model better insight into the defect-proneness of modules than the OOMs for a significant number of projects.

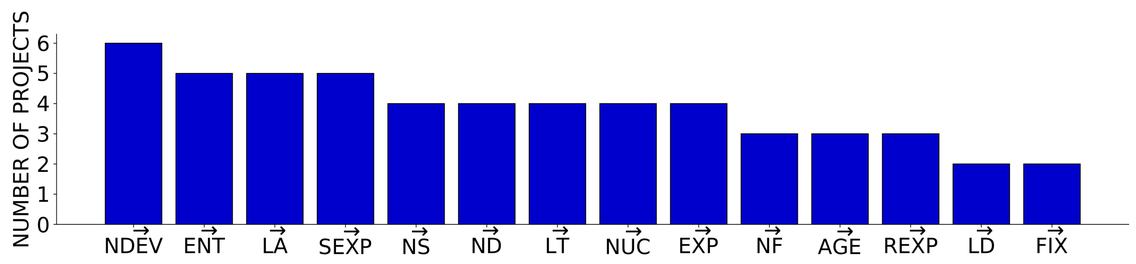

Figure 3. Number of projects where ACMs were selected among the 20 most important features for RFC.

DP-GCNN achieved a 13% higher average F-score than the RFC build with modules represented by OOMs. Its performance in terms of average F-score is compared with the performances of state-of-the-art models CNN [Pan *et al.*, 2019], DBN [Wang *et al.*, 2016], DP-T [Zhang *et al.*, 2020], MPT [Shi *et al.*, 2021], and SEML [Liang *et al.*, 2019]. The result of the comparison, which can be seen in Figure 4, shows that DP-GCNN provides comparable results to these models for the analyzed projects.

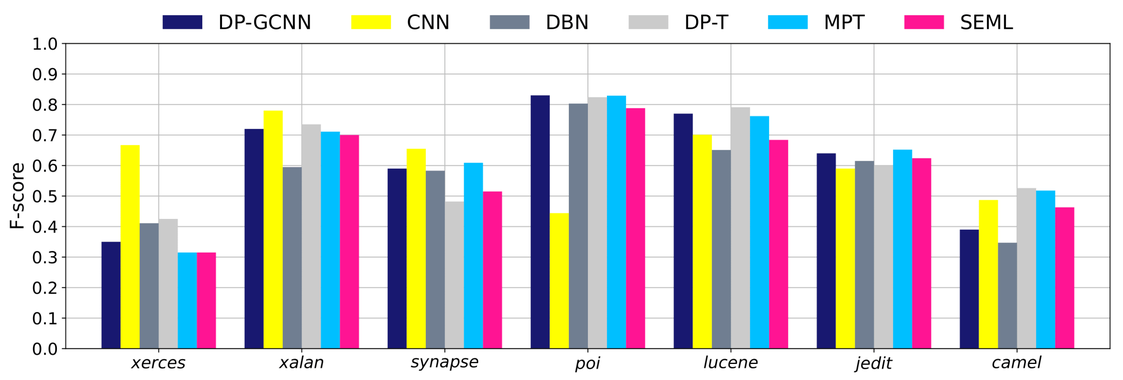

Figure 4. Performance comparison of DP-GCNN and state-of-the-art models in terms of F-score.

5. Conclusion

In the experiments conducted, it is shown that a commonly used binary classifier can achieve better results in predicting defect-prone files when the files are represented by metrics that accurately reflect their evolution process, rather than by the existing metrics. In addition, the results suggest that developing end-to-end models specifically tailored to graph structures, such as source code representations, can yield improvements in the area of software defect prediction. We believe that these results may inspire other researchers to explore similar approaches to achieve future improvements in software fault prediction.

*lucija.sikic@fer.hr