A Novel Approach for Code Smells Detection Based on Deep Leaning

Tao Lin[1], Xue Fu[2], Fu Chen[3], Luqun Li[4]

[1] Amazon, Seattle WA 98109, USA, paper@Ltao.org
[2] Shanghai Normal University, China
[3] Central University of Finance and Economics, China, chenfu@cufe.edu.cn
[4] Shanghai Normal University, China, success@shnu.edu.cn

Abstract. Compared to software bugs, code smells are more significant in software engineering research. It is not easy to detect code smells through traditional methods. In this work, we propose a novel code smells detection approach based on deep learning. The experiments show that our work achieves high scores in terms of F2 score.

Keywords: Code Smells, Deep Learning, Convolutional Neural Network.

1 Introduction

Code smells are increasingly generated by modern agile software development. This is because code changes are much more frequent and occur on a daily basis for large software companies and dominant open-source communities.

Although there are many more test approaches to detect code smells, these methods have some defects. Due to the frequent changes, it is increasing probable to generate code smells overheads. Code smells, like software bugs, are a serious problem in modern software.

Nowadays, the code smells are being researched by many practitioners. Software developers are not aware of what is the code smell, although they are aware of software bugs, thanks integrated environment development kits that can provide many instant suggestions and notifications when there are bugs.

The question here is how to detect code smells effectively? And what is the motivation for detecting code smells? Although there are many more test approaches to detect code smells, these methods have some defects. There are mainly two categories of deep learning networks. One is Recurrent Neural Networks, and another is Convolutional networks.

Corresponding Authors: Tao Lin (Amazon, USA, paper@Ltao.org), Luqun Li (Shanghai Normal University, China, success@shnu.edu.cn), Fu Chen (Central University of Finance and Economics, China, chenfu@cufe.edu.cn), Xue Fu (Shanghai Normal University, China)

Convolutional networks have already demonstrated its usage by leveraging hierarchy features. In this paper, we use fully convolutional networks for code smells detection based on semantic features. We will use fully convolutional networks for this work.

We will define the type of neural network we use, and explain how it is used to detect code smells. An advantage of using convolutional network is its ability to identify and use local correspondences.

In recognition and machine learning, convolutional networks are increasingly significant. Convolutional network presents the improvement on image recognition. An example is using convolutional network on local correspondence. In software engineering, we definitely can use these kinds of information for code smells detection.

To our knowledge, this is the first work to train a convolutional network for code smells recognition. The inference is much more improved through convolutional network.

This work is an extension of the authors previous work [8].

Unlike previous works that needs additional information for code smells detection, this work does not use any existing information for code smells detection. One of the major challenges in code smells detection is to find the relationship between code semantics and code location. There is a tradeoff between identifying the correct semantics compared to identifying the correct location of the smells.

Although there are several success stories from image recognition by using deep networks [1].It is hard to transfer these approaches to software engineering, which is more deterministic. Fully convolutional network has been used for one-layered computation, and has a potential to be deployed to multi-layered environments.

2 Code Smells Detection Based on Convolutional Networks

We can define a multi-dimensional array to represent the convolutional network, h * w * d, where h and w are space dimensions, and d is the channel. The first layer is our source code inputs.

The second layer is the networks for sequence modeling. For example, the inputs are x_0, x_1, x_2, x_3, x_4,… x_n, and the outputs are y_0, y_1, y_2, y_3, y_4,… y_n. The second layer will be y'_0, y'_1, y'_2, y'_3, y'_4,… y'_n.

The outputs will be reshaped to a one-dimensional array, where size will be D *1024. This output array will be dilation blocks. For the encoder task, we should process noise. Each layer in the encoder is processed by normalization and liner analysis.

3 High level design

With recent development in software engineering, it is easy to find software bugs using several methods from compiling to running. Our work is based on the state-of-the-art deep learning methods for detection and recognition task.

Firstly, we transform the software source code to XML file, in order to be processed by deep learning models [2]. Then there are two steps: code segment proposal and classification. The code segment proposal leverages the heuristic search to generate following inputs. These segments are processed by CNN classifier. We try to avoid to use R-CNN, otherwise. One of the main reasons is that R-CNN uses selective search algorithm, which is time consuming.

We use the following equation for segments:

$$\text{Seg} = \left(\max rp^{2^{l}/_{D}} + minrq\right) * (I/D)$$

Seg is the segments, r is the rule of limits, and p is process variable, q is the next graph inputs, I is interception, and D is next destination.

3.1 Experiments results

We use an open-source database published by the authors' previous work[11].

The experiments results are shown as following table:

Table 1 Experiments results

	Precision	Recall	F-Score	Kappa
Long Method	0.528	0.674	0.754	0.635
Lazy Class	0.624	0.678	0.613	0.632
Speculative Generality	0.712	0.734	0.689	0.643
Refused Bequest	0.698	0.701	0.711	0.677
Duplicated code	0.543	0.568	0.594	0.585
Contrived complexity	0.783	0.792	0.810	0.802
Shotgun surgery	0.597	0.596	0.501	0.601
Uncontrolled side effects	0.801	0.799	0.805	0.810

From Table 1, this work achieves high performance in terms of F2 score, especially for the category of uncontrolled side effects and contrived complexity.

4 Conclusion

In this work, we conducted a research for code smells detection based on deep learning.

Our solution uses convolutional neural network for training a model to detect several common code smells problems in software engineering. The solution achieves satisfied F2 score with the average above 0.75.

Acknowledge

Part of this work is from the author's PhD study [1], before the author joining Amazon. Professor Fu Chen from Central University of Finance and Economics provided many constructive suggestions and perspectives for this work during author's PhD study. Professor Fu Chen and this work was supported in part by National Science Foundation of China under No.61672104.

Reference

[1] T. Lin, "A Data Triage Retrieval System for Cyber Security Operations Center," *Pennsylvania State Univ. Thesis*, 2018.

[2] T. Lin, "A Container - Destructor – Explorer Paradigm to Code Smells Detection," *J. Chinese Comput. Syst.*, vol. 37, no. 3, 2016.

[3] T. Lin and X. Fu, "Flame Detection Based on SIFT Algorithm and One Class Classifier with Undetermined Environment," *Comput. Sci.*, vol. 42, no. 6, 2015.

[4] T. Lin, C. Zhong, J. Yen, and P. Liu, "Retrieval of Relevant Historical Data Triage Operations in Security Operation Centers," in *From Database to Cyber Security*, Springer, Cham, 2018, pp. 227–243.

[5] T. Lin, "A Novel Image Matching Algorithm Based on Graph Theory," *Comput. Appl. Softw.*, vol. 33, no. 12, 2016.

[6] T. Lin, "Graphic User Interface Testing Based on Petri Net," *Appl. Res. Comput.*, vol. 33, no. 3, 2016.

[7] T. Lin, "A Novel Direct Small World Network Model," *J. Shanghai Norm. Univ.*, vol. 45, no. 5, 2016.

[8] T. Lin, J. Gao, X. Fu, and Y. Lin, "A Novel Bug Report Extraction Approach," in *International Conference on Algorithms and Architectures for Parallel Processing*, 2015, pp. 771–780.

[9] C. Zhong, T. Lin, P. Liu, J. Yen, and K. Chen, "A cyber security data triage operation retrieval system," *Comput. Secur.*, vol. 76, pp. 12–31, 2018.

[10] T. Lin, "Deep Learning for IoT," 39th IEEE -- International Performance Computing and Communications Conference, 2020.

[11] T.Lin, "Security Operations Center Retrieval," 2021. https://github.com/ltaocs/SecurityOperationsCenterRetrieval.