Improving Software Testing Processes Using Artificial Intelligence: The Case of Code Clone Detection

I. General Introduction

The project presented in this report aims to address the issue of code clones in software. Code clones are similar or identical code fragments found in different parts of a program. They can pose significant challenges in terms of software maintenance and evolution because modifications made to one fragment often need to be manually replicated in its clones, thus increasing the risk of errors and the workload for developers.

II. Problem of Code Clones

Code clones are identical or similar code fragments that can appear for various reasons such as code reuse or lack of time. Although clones may seem like a quick solution, they can make the code difficult to maintain in the long term, as changes must be made consistently across all clones.

Moreover, clones can mask bugs and security vulnerabilities, as a problem present in one clone can be found in all the others.

There are tools and techniques to detect and manage code clones, such as static analysis and refactoring. In general, it is recommended to avoid code clones as much as possible by using good development practices such as code reuse through libraries and frameworks, abstraction, and modularization. This improves code quality, facilitates maintenance, and reduces the risk of errors and security vulnerabilities.

III. Approaches Used

To tackle this issue, several approaches were implemented in this project:

Use of language models: Models such as CodeBERT and Universal Sentence Encoder (USE) were used to encode code fragments into feature vectors.

Siamese neural network: A Siamese neural network architecture with a contrastive loss function was set up to enhance the differentiation between cloned and non-cloned code fragment pairs.

Machine learning algorithms: Algorithms like Support Vector Machines (SVM), Random Forests (RF), and XGBoost (XGB) were used for the classification of code clones.

1. Methodology

Preprocessing and Tokenization: Source code data was preprocessed and tokenized using WordPiece, allowing the division of code sequences into individual tokens.

Encoding: Code fragments were encoded into feature vectors using language models such as CodeBERT and USE.

Learning Models: The encoded vectors were used as input for various machine learning models, including SVM, RF, and XGB, to classify code fragments as clones or not.

2. Model Performance

The performance of the models was evaluated in terms of precision, recall, and F1 score. Here is an overview of the results obtained:

Model	Precision Score	Recall Score	F1-Score
USE+XGB	71%	85%	77%
CodeBERT+XGB	73%	88%	79%
SNNCodeBERT	**76%**	**91%**	**82%**
RtvNN	20%	91%	33%
CDLH	74%	92%	82%
FA-AST-GMN	**94%**	**96%**	**95%**

IV. Results and Conclusion

The results obtained show that the proposed approaches are effective in detecting code clones. In particular, the approach based on the Siamese neural network combined with CodeBERT showed remarkable performance, surpassing other methods in terms of precision, recall, and F1 score. The FA-AST-GMN method, using Graph Neural Networks (GNN), also showed superior performance but requires more in-depth expertise and more delicate handling of code structures.

In conclusion, the advanced artificial intelligence techniques used in this project have demonstrated their potential to improve the detection of code clones, thus contributing to the efficient maintenance and evolution of software.

V. References

- Hou Min et Zhang Li Ping. “Survey on Software Clone Detection Research”. In : (2019)
- Wenhan Wang et al. “Detecting Code Clones with Graph Neural Network and Flow-Augmented Abstract Syntax Tree”. In : (2020)
- Hui-Hui Wei et Ming Li. “Supervised Deep Features for Software Functional Clone Detection by Exploiting Lexical and Syntactical Information in Source Code”. In : (2017).
- Martin White et al. “Deep Learning Code Fragments for Code Clone Detection”. In : Journal (2016)
- Zhangyin Feng. “CodeBERT : A Pre-Trained Model for Programming and Natural Languages”. In : (2020)
- Ruslan Salakhutdinov Gregory Koch Richard Zemel. “Siamese Neural Networks for One-shot Image Recognition”. In : (2017).
- Universal Sentence Encoder Daniel Cera, Yinfei Yanga, Sheng-yi Konga, Nan Huaa, Nicole Limtiacob, Rhomni St. Johna, Noah Constanta, Mario Guajardo-C´ espedesa, Steve Yuanc, Chris Tara, Yun-Hsuan Sunga, Brian Stropea, Ray Kurzweil
- https://github.com/clonebench/BigCloneBench