🔍

AI for detecting code plagiarism

☰ Tags

https://prod-files-secure.s3.us-west-2.amazonaws.com/dda62111-67f9-4bc2-8d24-91d9290e5fa9/9ef3a36e-c0cb-4b6a-9054-8102f5343e37/PMU-B_Coding_AI-Chaiyong_compressed.pdf

Problem Statements

- The existing techniques and tools are still facing challenges when detecting clones with several modifications (e.g., added/deleted/modified statements).
- Existing clone detection and plagiarism detection tools are difficult to use because it is command line based tool.

What are Code Clones?

Code Clones are two or more code fragments that are similar enough to be considered duplicates. They can be classified into two main types:

- **Syntactic clones** are identical or nearly identical code fragments, except for differences in layout, white space, and comments.
- **Functional clones** are code fragments that have the same functionality, even if they have different syntax or algorithms.

Code clones can be found in all types of software, and they can be caused by a variety of factors, including:

- **Copy-paste errors**
- **Reusing code from other projects**

- **Using code templates**
- **Automated code generation**

Modelling

1. Data Collection and Preparation:

- **Data Source:** BigCloneBench, a large and reliable dataset of code clones.
- **Data Splitting:** The dataset is split into training and testing sets using stratified sampling to ensure balanced representation of different clone types.

2. Code Metrics Extraction:

- **Syntactic Metrics:** 11 metrics are extracted, focusing on structural characteristics of the code (e.g., number of tokens, identifiers, operators, differences in file and method names, return types, lines of code).
- **Semantic Metrics:** 12 features are obtained using code2vec, a neural model that represents code snippets as fixed-length vectors capturing their semantic meaning.

3. Machine Learning Models:

- **Models Considered:** Decision Tree, Random Forest, and Support Vector Machine (SVM) with Sequential Minimal Optimization (SMO).

Key Points:

- **Data Quality:** Emphasizes the use of a reliable dataset (BigCloneBench) for training and evaluation.
- **Feature Engineering:** Combines both syntactic and semantic metrics to capture different aspects of code similarity.
- **Model Exploration:** Considers multiple machine learning models to find the best fit for the task.

Evaluation

Accuracy Evaluation:

- **BigCloneBench (BCB) Dataset:**
  - Precision, Recall, and F1-score values were calculated for different clone types.

    - The model achieved overall good performance, with F1-scores ranging from 0.65 to 0.86.
- **Real Software Projects:**
    - The model was evaluated on three real-world projects (JUnit4, Natty, and Merry).
    - Precision scores varied, ranging from 0.41 to 0.77.
    - Evaluation involved human experts to judge clone pairs.

Tool Adoption Evaluation:

- **Target Users:** Computer Science students, programmers, and developers.
- **User Study Methodology:**
    - Between-Subjects design: Participants were randomly assigned to either a command-line tool (Simian) or the Merry web-based tool.
    - Metrics: Likeliness of using the tool, ease of understanding, and ease of use.

Conclusions

Key points:

- **Merry is a web-based tool that uses machine learning to detect code clones.**
- **Goals:** Improve accuracy and user experience compared to existing tools.
- **Merry Engine:** Uses 4 machine learning models and shows promising performance on BigCloneBench dataset but has varied results on real projects.
- **Merry Web Application:** Integrates with GitHub and offers a user-friendly interface for code clone detection.

Challenges and Limitations:

- **Supports only Java language.**
- **code2vec performance issues.**
- **Evaluated performance might not reflect real-world projects.**
- **Database scalability concerns.**

- **Potential bias in user study due to small sample size.**

Future work

- Expand the tool to detect clones in other languages
- Improve the code2vec run-time
- Create dedicated machine learning model per clone type
- Solve the MongoDB limitation by query a part of MongoDB document at a time
- Expand the number of participants in the user study.