EAST WEST UNIVERSITY

Fall 2023

Project Title: Code Clone Seeker
Course Title: Software Engineering
Course Code: CSE412
Section: 3
Group: 4

Group Info:

Name	**ID**
Abu Zafor	2020-2-60-158
Khandaker Mushayed Rahman	2020-2-60-125
Kazi Towfiq Tasmin Ornov	2020-2-60-133
Shahriar Haque	2020-2-60-149
Fahim Shahriar	2020-1-60-146
Alifya Akhter	2020-1-60-042

DATE: 28/12/2023

Code Clone Seeker: A Comprehensive Exploration of Duplicate Code Detection Techniques

Fahim Shahriar
Dept.of Computer Science and Enginnering
EAST WEST UNIVERSITY

Towfiqur Arnob
Dept.of Computer Science and Enginnering
EAST WEST UNIVERSITY

Shahriar Haque
Dept.of Computer Science and Enginnering
EAST WEST UNIVERSITY

Alifya Akhtar
Dept.of Computer Science and Enginnering
EAST WEST UNIVERSITY

Md.Abu Zafor
Dept.of Computer Science and Enginnering
EAST WEST UNIVERSITY

Khandaker Mushayad Rahman
Dept.of Computer Science and Enginnering
EAST WEST UNIVERSITY

***Abstract*—In software development, duplicate code detection is a procedure used to find and remove code that is exactly the very similar inside a codebase. Code clones, or duplicate code, happen when identical or strikingly similar code fragments show up in several places within a software project. Duplicate code can result in a number of problems, such as harder maintenance, more errors, and lower-quality software overall.**

***Keywords*—code clonet, token, stop ward, Lemmatization, TF-IDF Vectorization**

INTRODUCTION :

Complex codebases is frequently created throughout the dynamic and collaborative process of software development. Developers may unintentionally produce redundant or duplicate code segments, also referred to as code clones, across different portions of a project in their pursuit of functionality. Duplicate code is a major source of problems for overall code quality, bug fixing, and software maintenance. The efficience and long-term viability of software systems depend critically on the identification and mitigation of this duplication. This paper explores the application of Natural Language Processing (NLP) and Machine Learning (ML) techniques to the intricate task of duplicate code detection. By harnessing the power of NLP, we aim to transcend traditional syntactic analyses, considering the underlying semantics and intent of code snippets. Leveraging ML models trained on extensive datasets, we endeavor to enhance the accuracy and effectiveness of duplicate code identification. For people who are interested in identifying and handling duplicate code in software, this paper is a valuable resource. It facilitates understanding of the tools that are available and their importance in software development.

Literature Review:

CLISTER   Duplicate code [1] is one of the most serious problems in the programming world. This paper proposed an algorithm which efficiently calculates the dissimilarity between two chunks of code.The algorithm they used in this paper is the Multi Level Group Detection. This algorithm divides the whole code in a bunch of chunks which is then compared using the AD Tree algorithm. The dissimilarity percentage is calculated using the entropy or GINI values. Asthe number of comparable group might be high they use efficient search algorithms to reduce the number of comparable groups which increases the accuracy of the algorithm. The papers also mention future work such as adopting the record hiding concept for better privacy, handling missing values, comparing different duplication techniques and also improve this algorithm so that it can be used in video, image and web documents for check duplication.

Code Clones: Detection and Management The paper provides a comprehensive overview of code clones, their detection, and management in software systems. It covers a variety of topics, including clone detection techniques, clone types, and clone management approaches. The authors hope to give academics a starting point to learn about the basic ideas, strategies, and general steps for detecting and managing code clones. They also highlight research problems in this field. The research compares various clone detection and management methods based on a variety of parameters. It provides a brief review of various clone kinds and existing methods for detecting them, such as the usage of suffix trees and hashing for comparison. It also goes into the stages that go into clone discovery, such as match detection and aggregation. Overall, the paper is a valuable resource for code clone detection and management researchers as it provides an overview of existing techniques and identifies areas for future research.

Comparison and evaluation of code clone detection techniques and tools. The paper provides a detailed exploration of methods and tools used to identify identical or similar code fragments within software. The methods and tools available for identifying duplicate codes have been categorized in this study. The paper not only classifies various techniques and resources but also highlights how important it is to find duplicate code. Coding repetition in various software components might cause issues.written by Neha Saini For example, a mistake in one code may be copied in other locations, making corrections difficult. Detecting and addressing these repeated code fragments can save time and prevent errors The paper describes tools that can identify these kinds of similarities. For people who are interested in identifying and handling duplicate code in software, this paper is a valuable resource. It facilitates understanding of the tools that are available and their importance in software development.

Duplicate code detection [2] using anti-unification wriiten by the Peter Bulychev. The research paper "Duplicate code detection using anti-unification and metrics" proposes a process to identify repeated code in software systems using anti-unification and metrics. This text talks about the issue of using the same code multiple time. Then it mentions about how it can cause problems in creating and managing software. There is also a discussion about the clone detection tools that already exist and explains what problems they have. After that, they talk about anti-unification and how to discover code duplication using it. They propose a three steps method to identify the repeated codes. At first, they looks for sentences that have similar information using a method called anti-unification. Next, they put all of these statements that are alike in groups. At last, they check how alike these groups are using specials measurements. The articles also talks about other tools that find copies of code and tells about the outcome of tests done on open-source projects.

Duplicate Code Detection using Control Statements [3] written by Sudhamni. Code duplication is a very serious problem. It adds newly styled code in an existing project which is sometimes hard to maintain. There are many types of duplicate code such as similar code except comment, or changes in variables. There can be changes in functionality also like iterative and recursive ways to solve a code. The first approach used in this paper is to find the similarity between the structure of the code using control structure architecture. The initial process is pre-processing where they remove extra spaces and comments. Then they compare CST's (Control Structure Tables) of both code and compare the differences between two tables. In the second approach they used the order of execution where they add a new table called Function Information Table. This table stores the information about the starting and ending point of the function in a Control structure table. The experiment results suggest that their approach outperformed other methods to identify duplicate code. They referenced other method in their paper such as string based, token based etc. They suggested that their work can be improved int he future. Overall the results of their proposed way to determine duplication in a code work great compared to other methods.

Visual Detection of Duplicated Code. "Visual Detection of Duplicated Code [4] by Matthias Rieger and Stéphane Ducasse, the authors address the issue of code duplication in software development. They highlight the causes of duplicated code, emphasizing its negative impact on software quality, including defects, code bloat, and design flaws. The central method involves creating a two-dimensional matrix by comparing each line of code with every other line, allowing for visual exploration and automatic evaluation of duplicate patterns. The DUPLOC tool is introduced, which compares code line by line, provides noise reduction, batch mode, and a map to record occurrences. While the tool has some drawbacks, such as reliance on syntactic-level comparison and a basic string matching method, it is seen as a valuable contribution to software engineering for exploring and visualizing duplicate code.

The Adverse Effects of Code Duplication in Machine Learning Models of Code written [5] by the Miltiadis Allamanis .In the background of huge code, the abstract provides a thorough inspection of the consequence of code duplication on machine learning models. It shows the significant warning that code duplication dissimulation to the perfection evaluation of machine learning models in the source code analysis. The study provides tools to help the research community direction the problem of code duplication in forthcoming studies, as well as best practices for gathering code collective and the introduction of a duplications index for widely used datasets. The examination reveal that performance metrics can be filled up to 100 percents when testing on duplicated code associated and underscoring the need to consider code duplication in machine learning researches on code. This also introduces a method for detecting near-duplicates in various programming languages, specifically code predictive text. It can be said that the theoretical outlines the key question of codes duplication in machine learning models of source code, emphasizing the need for researchers to consider this factor In additionally, the paper offers valuable insights and best practices to help mitigate the problem of code duplication in machine learning researches

Comparison and evaluation of code clone detection techniques and tools [6] A qualitative approach written by the Chanchal K. Roya. The paper provides a detailed exploration of methods and tools used to identify identical fragments within software. The methods and tools available for identifying duplicate codes have been categorized in this study. The paper not only classifies various techniques and resources but also highlights how important it is to find duplicate code. Coding repetition in various software components might cause issues. For example, a mistake in one code may be copied in other locations, making corrections difficult. Detecting and addressing these repeated code fragments can save time and prevent errors. The study also describes how these methods work, including checking for code that is similar or slightly different. For instance, imagine being able to tell the difference between two nearly identical statement containing a few different words. The paper describes tools that can identify these kinds of similarities. For people who are interested in identifying and handling duplicate code in software, this paper is a valuable resource. It facilitates understanding of the tools that are available and their importance in software developments.

Objectives:

- Develop a machine learning-powered duplicate code detection tool integrated into Google Colab through Streamlit.
- Design a model that accurately detects code duplication to similarity percentage
- Support multiple programming languages for broader applicability.

Methodology

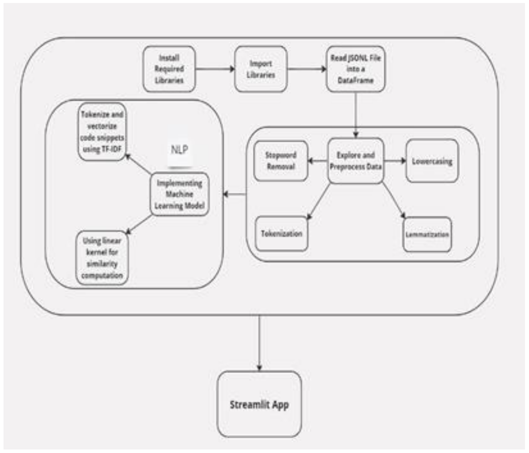
In our pursuit of advancing duplicate code detection methodologies, we leverage the Big Clone Bench dataset—an integral resource in the field of software engineering for evaluating and benchmarking clone detection tools. Big Clone Bench stands out as a comprehensive and diverse dataset designed to rigorously test the capabilities of code clone detection techniques. Here, we provide an overview of the key features and significance of Big Clone Bench in the context of our research. We use the Big Clone Bench dataset, which is a tool for software engineers to compare clone detection algorithm, in efforts to advance duplicate code detection methodologies. Big Clone Bench is a distinct dataset that is both extensively and varied, and it is specifically created

Dataset Description:

Big clone bench data set is related to duplicate code detection or code similarity analysis. This data set included various types of programming languages such as ,c++,Java, python, php etc. This dataset is used to designed to analysis research area of clone code detection**.** We collected this dataset from kaggle

Implementation:

–
.


Install Required Libraries
Import Libraries
Read JSONL File into a DataFrame
Tokenize and vectorize code snippets using TF-IDF
NLP
Implementing Machine Learning Model
Using linear kernel for similarity computation
Stopword Removal
Explore and Preprocess Data
Lowercasing
Tokenization
Lemmatization
Streamlit App

Pre-processing:

1.TOKENIZATION:

Code snippets are inherently unstructured and need to be broken down into meaningful units for analysis. Tokenization involves dividing the code into individual tokens, which can be as granular as words or as specific as programming language symbols. In our preprocessing pipeline, the NLTK library is employed for tokenization. This step facilitates a more fine-grained analysis of the code structure and content.

2. Lowercasing:

To ensure uniformity and avoid treating words with different casings as distinct entities, all tokens are convertes to lowercase. This step aids in standardizing the textual representation of the code.

3. Stop word Removal:

Common words that do not contribute significantly to the meaning of the code (e.g., "if," "else," "for") are removed to reduce noise in the analysis. The NLTK library provides a set of stopwords for various languages.

4. Lemmatization:

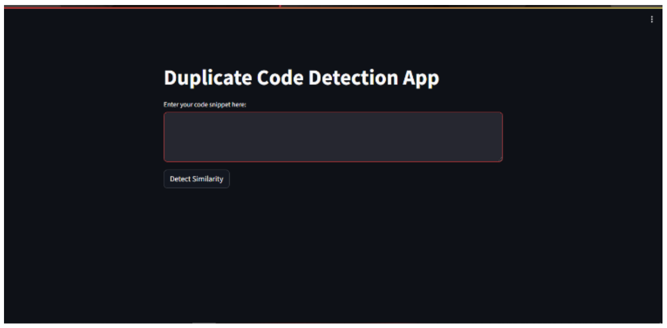
Lemmatization involves reducing words to their base or root form. This is crucial for collapsing variations of words into a common representation, enhancing the semantic understanding of the code. The WordNet Lemmatizer from NLTK is employed for this purpose.

TF-IDF Vectorization:

The final preprocessing step involves vectorizing the tokenized and lemmatized code snippets using the Term Frequency-Inverse Document Frequency (TF-IDF) technique. This converts the code snippets into numerical vectors, considering the importance of each term within the snippet relative to its frequency across the entire dataset. The Scikit-learn library provides an efficient implementation for TF-IDF vectorization.

Stream lit App:

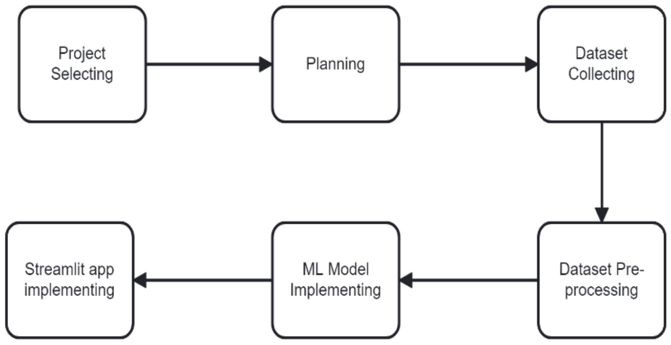
An open-source Python library called Stream lit is used for creating web applications data science and machine learning. Here is our streamlit app that we implemented where an user will give their code snippet and can show the similarity percentage of that code as well.



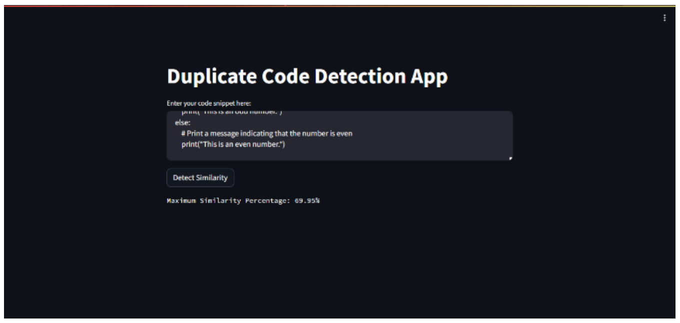
Duplicate Code Detection App

Enter your code snippet here:

Project working flow diagram:



Result and Discussion:



Duplicate Code Detection App

Enter your code snippet here:

```
    print("This is an odd number.")
else:
    # Print a message indicating that the number is even
    print("This is an even number.")
```

```
Maximum Similarity Percentage: 69.95%
```

The evaluation of code similarity methodologies yielded a maximum similarity score of 69.95% between an online code snippet and the dataset. This score indicates a substantial degree of similarity, suggesting shared logic or functional resemblance between the online code and the dataset's code snippets.

The 69.95% similarity score signifies a noteworthy alignment between the online code snippet and the dataset. This level of similarity implies significant structural and lexical resemblances, indicating potential duplicity or shared functionalities. However, the accuracy of the code is not accurate because of the lack of the dataset or may be for the performance of the model.

Future Works:

Our future plan is to collect a large dataset so that our model can work well. After that, we will try to use more advanced techniques For AST-based similarity or embeddings (like Word2Vec or Doc2Vec), we might need specialized libraries or approaches. For this, we have to do some studies about those specialized libraries so that we can know how to use them. We also want to work with source detection so that users can understand from which source the given code is copied and underline the copied code in the result.

Challenges and Consideration:

1. As we did not work with Natural Language Processing before so we have faced so many challenges to implement it

2. We have faced some difficulties to collect the datasets.

3. Explore techniques to handle diverse programming languages and syntax structures

4. As we did also mention about Streamlit app which was very new to us to implement. So, we had to study about the Streamlit app for implementation.

CONCLUSION

Clone code seeker plays a vital role in the field software development maintaince similarity. In this project, we implemented a clone code seeker that detects code similarity copied from other sources. Here we used the NLP model for similarity detection. Finally, we built a streamlit app for a better user interface. In this interface, a user can input the code snippets that are copied and can checks the similarity percentage. In the future world for the software engineers cannot copy the code and face the copyrights problem. To summaris, clone code seekers play a vital role in preserving a robust and effective software development process by enhancing code quality, minimising redundancy, and promoting improved teamwork among development teams

References

1. Baker, B. (1993). A program for identifying duplicated code. Computing Science and Statistics, 49–49.

2. Kumar, A., & Vengataasalam, S. (2013). CLUSTER BASED DUPLICATE DETECTION. Journal of Computer Science, 9(11), 1514.
C:\Users\USR\OneDrive\Desktop\paper\address
3. Saini, N., Singh, S., & others (2018). Code clones: Detection and management. Procedia computer science, 132, 718–727.

4. Roy, C., Cordy, J., & Koschke, R. (2009). Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. Science of computer programming, 74(7), 470–495.5.The Adverse Effects of Code Duplication Miltiadis Allamanis.

5. Bulychev, P., & Minea, M. (2008). Duplicate code detection using anti-unification. In Proceedings of the Spring/Summer Young Researchers' Colloquium on Software Engineering.

6. Sudhamani, M., & Lalitha, R. (2015). Duplicate code detection using control statements. International Journal of Computer Applications Technology and Research, 4(10), 728–736.

7. Hordijk, W., Ponisio, M., & Wieringa, R. (2009). Harmfulness of code duplication-a structured review of the evidence. In 13th International Conference on Evaluation and Assessment in Software Engineering (EASE) 13 (pp. 1–10).

9. Allamanis, M. (2019). The adverse effects of code duplication in machine learning models of code. In Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (pp. 143–153).

10. Rieger, M., & Ducasse, S. (1998). Visual detection of duplicated code. In ECOOP Workshops (pp. 75–76).

Google Colab Link:

https://colab.research.google.com/drive/1bDqFPFYwWUlgjRL8RK668dSo9NCbmgRf?usp=sharing