Approach to Plagiarism Detection in Programming Assignments

Malek Algabri[1,2], Firdaus Alhrazi[1]

[1]College of Computer Science and Technology, Sana'a University

2College of Engineering and Information Technology, Emirates International University, Sana'a, Yemen

malekye@su.edu.ye, dr.malekye@eiu-edu.net , firdaus.mansoor2024@gmail.com

Abstract:

People tend to shortcut ways that save them time and effort to do the tasks required by them, either by taking tasks ready-made online, or stealing someone's work as their own. Since everything now is connected to the Internet, there is a very high potential of duplicating or stealing someone else's work, which is known as plagiarism. With the advancement of technology, it has become quite simple to do all tasks through the Internet. Plagiarism is the copying of other people's ideas and actions; it is considered a crime. Plagiarism occurs due to laziness, fear of failure, and the desire to perform the required tasks without fatigue or effort. In this paper, a methodology for detecting plagiarism in programming tasks, in particular in the visual programming category using deep learning and machine learning algorithms is proposed. Also, a solution has been proposed to detect plagiarism in the source code and interfaces that pertainto programming assignments.

Keywords: Index Terms Plagiarism Detection, Programming Assignments, Machine Learning, CNN, MILEPOST GCC.

I. INTRODUCTION

Plagiarism detection is useful in the era of technological development. Plagiarism detection helps to maintain integrity in the academic field, as it is easy to copy or steal the worksof others via the Internet. Most people tend to steal or copy someone's work to save effort and time in getting the costsor work required of them done. If people continue to stealthe works of others, the original work will not be valuable at all. Plagiarism in programming tasks is a big problem, where students copy codes from their classmates, or find codes onlineand utilize them without indicating to original code [1].

There are many plagiarism detection tools, and these tools have greatly reduced manual checking. Most plagiarism de- tection tools use syntactic-based methods and a text-based ap- proach [2]. These tools tend to fail when the code obfuscation process is done. In programming tasks, code obfuscation can be done by changing variable names, rearranging sentences, replacing loops and control sentences, converting variables, and adding comments.

It is very difficult to identify copied work in programming languages because there are different types of plagiarism, which are difficult to find with plagiarism detection tools.Tools like JPlag [3] and MOSS [4] remove comments before testing for plagiarism, which can lessen their importance for plagiarism detection. There are different types of plagiarism

[5] such as direct plagiarism, smart plagiarism, self-plagiarism, and unintentional plagiarism. Direct plagiarism is copying without any change or quotations. Smart plagiarism is when someone plagiarizes an original work skillfully, they rephrase sentences in their way. Self-impersonation is when someone presents the same as their previous work in other situations andtasks. Unintentional plagiarism where this type occurs when someone inadvertently reformulates a topic [5].

In this paper, the focus will be on plagiarism detection of source code and interfaces in visual programming class. The goal is to solve the problem of plagiarism in students' softwaretasks.

II. RELATED WORK

Most universities use online code evaluation platforms [6], making plagiarism an easy task in programming tasks. Imper- sonation in general in the field of education is considered a betrayal of academic honesty and a moral crime. There aretwo ways to detect plagiarism [5]:

- Document plagiarism.

- Source code plagiarized.

A. *Existing Tools*

- Tools for detecting plagiarism online:

  An online plagiarism detector is a tool that takes text or documents as input and checks or compares them with several articles and papers published that have been published online. It produces a report at the end for reference that shows the proportion of information that has been plagiarized [5].

- Independent tools for detecting plagiarism:

  Unlike online plagiarism services, plagiarism detection apps may be installed and used on individual PCs andare used to check for plagiarism in specified input textsor documents. To find text matches, it does a huge amountof internet searches using articles. For these resources to work, the device must be connected to the Internet [5].

B. *Plagiarism Detection for Source code and Texts*

Based on a survey [7] of computer undergraduate students, 54% admitted plagiarizing and over 87% reported that their assignment was plagiarized. According to article [8] students cheat for a variety of reasons, including the ease with which

they may access or exchange material online, the low like- lihood of being discovered, a lack of time, a fear of being judged, and so on. As a result, when students complete the course, they are deficient in important knowledge, and their entitlement to a fair and equal evaluation of their expertise is denied. Due to the large number of students in each course, educators rely on assignments to determine which students need further help and whether the course can be made better [6].

C. *Detection of Source Code Plagiarism*

Source code plagiarism occurs when someone copies or reproduces the same source code without properly citing the original source code author [5]. There are two methods available:

- Syntaxial Change

  The shift is the simplest form in syntaxial. They may be completed by making a few straightforward adjustments to the main code. In this situation, knowing how to code is not necessary [5].

- Structural Change

Changes to calls made within the process body and vice versa, changes to iterations, conditional statements, and statement order, as well as the conversion of procedures to works and the inclusion of statements that will not modify the code's output are all instances of structural changes [5].

D. *Process Model for Plagiarism Detection*

An enhancement to the process model used to identify plagiarism. Five phases make up the comparable model seenin [1]:

- By tokenizing the source code, detection becomes re- sistant to changes in the source code's structure, likelanguage translations and variable renaming.
- Pre-processing improves the detection of source code changes such as removing comment blocks, splitting or merging variable declarations, rearrangement of state-ments, etc.
- deleting and creating a tokenized version of source code that is allowed.
- Utilizing similarity measurement to find similarities.
- Finally, calculating the amount of similarity between two source codes.

E. *Detecting Source Code Plagiarism Bytecode Approach*

A technique for detecting source code plagiarism that uses low-level instructions instead of source code tokens may catch the majority of plagiarism assaults from beginner pro- gramming courses at any level. Additionally, a number of techniques, including instruction reinterpretation, instruction generalization, method linearization, and method-based com- parison are incorporated to increase its efficacy. The choice ofJava as the targeted programming language with Bytecode as its low-level instruction was made, due to low-level instructionbeing a language-dependent characteristic based on analysis

[9]. Most source code plagiarism detection tools are created with language independence in mind. This kind of approach generalizes all computer languages and treats them as raw text,assuming that the most obvious plagiarism indications are not language-dependent [10]. Suggest a controlled experiment to gather plagiarism assaults because more accurate plagiarism detection may be built if the most likely plagiarism attacks are identified and reported. Based on 378 source codes created by respondents who stole work, we investigate potential plagia- rism assaults in a controlled experiment. Respondents are only allowed to be skilled programmers from different courses in order to gather more diverse plagiarism attacks. Additionally, the majority of them are trained to spot plagiarism in students'homework as lecturer assistants.

F. *Source Code Plagiarism Detection using Machine Intelli- gence Approach*

There are several methods available to assist preserve the requisite integrity and identify plagiarism. where it deals with plagiarism in a particular category of C programming tasks. The benefits and drawbacks of various deep learning and Machine Learning techniques are thoroughly examined. To identify plagiarism in source code, algorithms like SVM, KNN, RNNs, D-Trees, and attention-based transformer net- works are put to the test. During the course of this investigation, a large dataset made up of code pairings was created. The results collected demonstrate that the state-of-the-art text-based plagiarism detectors used today are not as accurate at identifying plagiarism as machine learning and deep learning approaches [2].

The authors in [11] use feature-based neural networks to try to detect plagiarism. They analyze the usefulness of the various characteristics for identifying plagiarism and use a neural network approach to detect plagiarism. The similarity provided by MOSS is employed as one of the characteristicsof their method and is also demonstrated to have the highest significance.

The authors in [12] go through various Machine Learning approaches to plagiarism detection. The outcomes of several experiments are used to illustrate them. Result of testing several machine learning techniques on feature pairings, conclusions are drawn.

III. METHODOLOGY

To detect plagiarism in source code and interfaces in visual programming, a methodology consisting of steps is proposed, which are as follows:

A. *Dataset Preparing*

There is the study [2] was done before the dataset is created. conducted to create a dataset containing pairs of plagiarized and non-plagiarized programs, and different methods were used to obfuscate the source code, and then collect programs from various tutorials and online platforms. In this section, the dataset will be used in the study [2], and data augmentation will be introduced.

Here, different techniques for code obfuscation were utilizedto provide a sufficient dataset that could accommodate a range of potential instances of source code and interfacetheft. Programs were gathered from schoolwork assignments,various tutorials, and from internet coding platforms to includea range of source codes and interfaces. Out of them, 519 pairsincluded plagiarism, whereas the remaining 722 pairs did not.Pair of copied source codes and interfaces that had been obfuscated using well-known methods. In order to make thedataset impenetrable against a range of problem sets, it wasmade sure that a variety of strategies were incorporated where

the different code obfuscation methods are as follows:

- Change in Data Types.
- Change in Variable Name.
- Changing from For Loop to While Loop.
- Rearranging the Statements.
- Coding Block Reordering.
- Expression Reordering.
- If Else converted statements.
- Change colors in interfaces.
- Change the order of tools in interfaces.

To detect plagiarism in visual programming, we must take two approaches. First, plagiarism detection in source code. Second, plagiarism detection in the interfaces. Below are more details.

B. *Textual Feature Extraction for Source Code*

In this approach, plagiarism of source code will be detected by extracting features from source code using machine learn- ing. The following is the plagiarism detection mechanism.

To extract features from a source code MILEPOST GCC with O3 was utilized [13]. A machine learning compiler calledMILEPOST GCC is used for testing and research. The scaleof each characteristic ranged from zero (the least value) to one(the greatest value).

In [2] Support Vector Machine (SVM), K Nearest Neighbors (KNN), and Decision Trees were the classification techniques employed on the condensed dataset. The next sections show how these algorithms are implemented and the accuracy that may be attained using them. The weights assigned based on the distance between the 7 nearest neighbors were taken into account when the K Nearest Neighbors algorithm was developed. The Gini Index served as the foundation for the Decision Trees Classifier method that was used. The maximum depth was 4, and the minimum number of leaves for the sample was five. Support Vector Machine (SVM) was the most recent algorithm used. The Grid Search Technique was used to get the SVM model's ideal parameters[14].

C. *Image Feature Extraction for Interface*

In this approach, plagiarism of interfaces will be detected by extracting features from interfaces as images using a Convolutional Neural Network (CNN). The following is the plagiarism detection mechanism.

When using traditional feed-forward neural networks to handle picture classification issues, image pixels might be

utilized directly as input. Compared to conventional fully connected neural networks, CNN networks are significantly quicker and more reliable [15]. The CNN model has excelled in a variety of computer vision tasks, including image process-ing. Additionally, we may utilize the CNN model to extractthe features. pull features from various network levels for use in other tasks. In general, global feature extraction based on CNN consists of feature extraction, pretraining the CNN model, and fine-tuning the CNN model [16].

The feature extraction pooling layer and convolution layer are two different types of network layers in the CNN model. To extract various picture information, the convolution layer convolves input data using several convolution kernels. The pooling layer samples the data coming in. The activation function then nonlinearly abstracts input information. After entering the CNN model, the original picture moves ahead through pooling, multi-layer convolution, and non-linear addressing. Additionally, the picture data is continuously abstracted. The final output, characteristics increasingly abstract from low-level semantic data to local specifics [16]. As a result, the CNN model has more low-level data and higher-level semantic information the closer it is to the input convolution layer and full connection layer, respectively.

In this step, the focus will be on the detection of plagiarism in the software interfaces, as students may perform plagiarism operations for the software interfaces, the solution is to use CNN so as to detect plagiarism between the interfaces. Through CNN the features of the interfaces will be extracted, and then the features extracted from the interfaces to be detected for plagiarism will be compared with the dataset createdin advance in the first step. The result of this comparison will be calculated as a percentage of the amount of differencebetween the interfaces to be examined and the previously prepared dataset.

IV. RESULT AND DISCUSSION

Using machine learning and deep learning, an algorithm was created to identify interface and source code plagiarism. By contrasting them and determining the proportion by utilizing the Machine Learning algorithm and CNN, we can determine the amount of plagiarism that has been done in visual pro-gramming.

Figure. 1 shows the performance of our method when compared to MOSS and JPlag. The blue bars represent the performance of our method, and the red bars represent MOSS and yellow JPlag.

We can find the amount of plagiarism performed on a pro- gram by comparing the source code and interfaces in the pro- gram and calculating the amount of percentage of the amount of difference between the program and the programs stored in the dataset prepared for this task regarding the source code,the percentage of difference is calculated by Machine Learning and the percentage of the difference between interfaces by CNN's deep learning algorithms. Plagiarism is verified by taking outputs and calculating the average percentages to find the final plagiarism percentage. After training on source code data [2], the results were tested on a test suite including 496 programs, which was processed into 248 pairs. The top levelof accuracy was obtained using the SVM model. The details obtained for each model are given in Table. I.

Model	Accuracy
SVM	97.42%
D-Trees	96.35%
KNN	96.95%
Proposed Work	98.95%

TABLE I

ACCURACY OF CLASSIFIER MODELS

A simple Convolutional Neural Network has also been pro- posed to detect plagiarism in the interface by classifying images. This was done by taking the features of the interfaces andcomparing them with the data in the dataset; then calculating all the proportions of differences between the data stored in the dataset and the interface that is required to be examined, and after that calculating the average of differences between those interfaces. In addition, in this methodology, interfaces are treated as images. This simple Convolutional Neural Network imposes a lower computational cost as well as better results, which is what distinguishes the methodology in this paper compared to plagiarism detection tools.

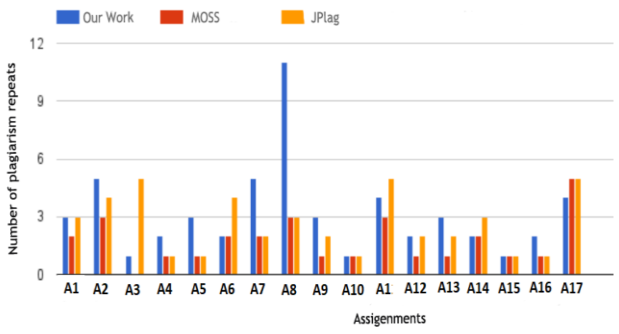

Figure I

Performance of our method when compared to MOSS and JPlag

V. CONCLUSION AND FUTURE WORK

Academic dishonesty is defined as plagiarism. Even though stealing someone else's work is legal, it may still violate their copyright. It is a serious, unethical infraction in academia. The law does not penalize plagiarism. Plagiarism is defined as using someone else's words or ideas without their permission or referencing the actual author of the material in order to pass it off as your own. This study was done on the application of deep learning and machine learning algorithms for plagiarism detection shows significant promise. Compared to earlier techniques, these new algorithms are far more potent. There is an opportunity to strengthen the algorithms, though. Future promises may include creating a tool that uses machine learning methods to show precisely where code noise occurs. Utilizing the Bag of Words algorithm to identify plagiarism is a further research field. There may be further techniques to obtain dynamic features from the program that can spur more advancement in this particular field and lengthen the useful life of plagiarism detection.

REFERENCES

1. Kermek, D. and M.J.I.i.E. Novak, *Process model improvement for source code plagiarism detection in student programming assignments.* 2016. **15**(1): p. 103-126.
2. Eppa, A. and A. Murali. *Source Code Plagiarism Detection: A Machine Intelligence Approach*. in *2022 IEEE Fourth International Conference on Advances in Electronics, Computers and Communications (ICAECC)*. 2022. IEEE.
3. Prechelt, L., G. Malpohl, and M.J.J.U.C.S. Philippsen, *Finding plagiarisms among a set of programs with JPlag.* 2002. **8**(11): p. 1016-.
4. Schleimer, S., D.S. Wilkerson, and A. Aiken. *Winnowing: local algorithms for document fingerprinting*. in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. 2003.
5. Yasmeen, S., et al. *Plagiarism Detection for Source Codes and Texts*. in *Sentimental Analysis and Deep Learning: Proceedings of ICSADL 2021*. 2022. Springer.
6. Alexandra-Cristina, C. and A.-C. Olteanu. *Material survey on source code plagiarism detection in programming courses*. in *2022 International Conference on Advanced Learning Technologies (ICALT)*. 2022. IEEE.
7. Sraka, D. and B. Kaucic. *Source code plagiarism*. in *Proceedings of the ITI 2009 31st international conference on information technology interfaces*. 2009. IEEE.
8. Albluwi, I.J.A.T.o.C.E., *Plagiarism in programming assessments: a systematic review.* 2019. 20(1): p. 1-28.
9. Karnalim, O. *Detecting source code plagiarism on introductory programming course assignments using a bytecode approach*. in *2016 International Conference on Information & Communication Technology and Systems (ICTS)*. 2016. IEEE.
10. Brixtel, R., et al. *Language-independent clone detection applied to plagiarism detection*. in *2010 10th IEEE Working Conference on Source Code Analysis and Manipulation*. 2010. IEEE.
11. Engels, S., V. Lakshmanan, and M. Craig. *Plagiarism detection using feature-based neural networks*. in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*. 2007.
12. Katta, J.Y.B., *Machine learning for source-code plagiarism detection*. 2018, International Institute of Information

Technology Hyderabad, University of ....

13. Fursin, G., et al. *MILEPOST GCC: machine learning based research compiler*. in *GCC summit*. 2008.
14. Platt, J.J.A.i.l.m.c., *Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods*. 1999. **10**(3): p. 61-74.
15. Li, Q., et al. *Medical image classification with convolutional neural network*. in *2014 13th international conference on control automation robotics & vision (ICARCV)*. 2014. IEEE.
16. Peng, X., et al., *Research on image feature extraction and retrieval algorithms based on convolutional neural network*. 2020. **69**: p. 102705.