IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 6, June 2024

DOI: 10.17148/IARJSET.2024.11649

# ONLINE ASSIGNMENT PLAGIARISM CHECKER

**Srijith C[1], Dr. Biju Balakrishnan Ph.D. [2]**

II MCA Students, Master of Computer Applications, Hindusthan College of Engineering and Technology, Coimbatore, India[1]

Assistant Professor, Master of Computer Applications, Hindusthan College of Engineering and Technology, Coimbatore, India[2]

**Abstract:** Plagiarism is defined as to take or theft some work and present it has one's own work. This plagiarism checker system is used to analyze the plagiarism data. Plagiarism affects the education quality of the students and thereby reduces the economic status of the country. Plagiarism is done by paraphrased works and the similarities between keywords and verbatim overlaps, change of sentences from one form to other form, which could be identified using Word Net etc. This plagiarism detector measures the similar text that matches and detects plagiarism. Internet has changed the student's life and also has changed their learning style. It allows the students to get deeper in the approach towards learning and making their task easier. Many methods are employed in detecting plagiarism. Usually plagiarism detection is done using text mining method. In this plagiarism checker software, user can register with their basic registration details and create a valid login id and password. By using login id and password, students can login into their personal accounts. After that students can upload assignment file, which will further divide into content and reference link. This web application will process the content, visit each reference link, and scan the content of that webpage to match the original content. Also, students can view the history of their previous documents.

**Keywords:** Plagiarism, Assignment, Werkzeug, Jinja,Flask.

## I. INTRODUCTION

Plagiarism is described as taking or stealing someone else's work and presenting it as one's own. The plagiarism data is analyzed using this grammar and plagiarism checker system. Plagiarism has a negative impact on student education and, as a result, the country's economic standing. Plagiarism is committed through paraphrased works, keyword and verbatim overlaps, and the conversion of phrases from one form to another, all of which can be detected using WordNet.

This plagiarism detection tool compares similar content and finds plagiarism. The internet has altered the lives of students as well as their learning styles. It allows pupils to get deeper into their learning approach while also making their task easy. Plagiarism can be discovered in a number of ways. Text mining is the most frequent method for identifying plagiarism. This plagiarism detector software allows users to register with their basic registration information and create a valid login id and password. By inputting their login id and password, students can access their personal accounts. The assignment file, which is divided into content and reference links, can then be uploaded by students. This web software will scan the content, parse it, and visit each reference link, comparing the content of that webpage to the original. Students might also research the background of previous documents. Students can also check the content for grammar errors.

## II. RELATED WORK

A.Plagiarism detection in computer programming using feature extraction from Ultra-Fine-Trained Repositories.

Author: VedranLjubovic and Enilpajic
Year: 2023

Overview: This study proposes a solution to these issues by tracking all activities that students engage in while working on assignments in a cloud-based integrated development environment (IDE). This log is handled like a source coderepository, and repository mining techniques are used to create "developer profiles," allowing suspect behavior to be detected.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 6, June 2024

DOI: 10.17148/IARJSET.2024.11649

The paper is focused on plagiarism detection, but a similar approach could be used for other purposes in an educational setting, such as detecting students who might need special attention or additional instruction in certain areas. Low resolution is a common issue with source repositories. The authors of this work propose leveraging the IDE "autosave" feature to record tiny changes to create an ultra-fine-grained repository, down to individual keystrokes. ‘‘Autosave’’ is usually implemented in a way that reduces the impact to system performance, such that a document is saved after a certain period of inactivity. File system monitoring tools are then used to automatically commit these changes to a Subversion (SVN) shadow repository that is invisible to the user. This allows researchers to use well-known tools for analysis and debugging of SVN repositories. This approach is also compatible with all known IDEs and editors that support autosave. Further, since ‘‘Run program’’ and ‘‘Test’’ buttons also result in the creation of various output fifties (executable, core dump, etc.), these can be analyzed to detect IDE interactions and log unit test results. The method proposed in this paper is a variation of Change Recording or Change Logging, but instead of inferring semantics of change, the system records concrete, atomic changes to the source code text, with the expectation that this high-resolution log will be analyzed further using machine learning methods. To demonstrate the usefulness of this approach, ultra-fine-grained repositories were created for 300 students working on programming homework in an introductory university programming course. These repositories yielded a set of features that were utilized to improve plagiarism detection. This paper's hypothesis is that students who plagiarism coursework use their IDE differently than students who work independently. Note that the solution presented here is cloud-based, so no tools were installed on student computers, especially tools that would monitor activity outside the IDE code editor window (i.e. keylogger), since that would be unethical. Students were asked to sign a consent form allowing their anonymized usage history to be utilized for research purposes, and those who refused were omitted from the dataset.

B. Online Assignment plagiarism checking using Data mining & NLP

Author: Taresh Bokade, TejasChede, DhanashriKuwar, Prof. Rasika Shintre
Year: 2021

Overview: Plagiarism is described as taking or stealing someone else's work and presenting it as one's own. The plagiarism data is analysed using this grammar and plagiarism checker system. Plagiarism has a negative impact on student education and, as a result, the country's economic standing. Plagiarism is committed through paraphrased works, keyword and verbatim overlaps, and the conversion of phrases from one form to another, all of which can be detected using WordNet. This plagiarism detection tool compares similar content and finds plagiarism. The internet has altered the lives of students as well as their learning styles. It allows pupils to get deeper into their learning approach while also making their task easy. In order to detect something, many ways are used. The most common method for detecting plagiarism is text mining. Users can register with their basic registration details and create a valid login id and password with this plagiarism checker software. Students can access their personal accounts by entering their login id and password. Following that, students can upload an assignment file that is divided between content and reference links. This website will analyze the material, visit each reference link, and compare the content of that webpage to the original. Students can also look up the history of prior documents. Teacher also able to check the grammar mistakes on the content and semantical plagiarism.

## III. METHODOLOGY

Finding plagiarized parts of a assignment is very slow work for teachers. Even with a limited number of texts it relies on the teacher's ability to read and remember every submission. As the process of finding plagiarized parts in assignment is based on the teacher's ability to remember all that he or she has read, the results may be incomplete. Some clear cases of copy and paste may easily be overlooked. And since the workload cannot be shared between multiple assistants. Thus we are introducing system for checking Plagiarism in assignments.

- Finding plagiarized parts of a assignment is very slow work for teachers.
- Even with a limited number of texts it relies on the teacher's ability to read and remember every submission.
- As the process of finding plagiarized parts in assignment is based on the teacher's ability to remember all that he or she has read, the results may be incomplete.
- Some clear cases of copy and paste may easily be overlooked. And since the workload cannot be shared between multiple assistants
- Active internet connection required.
- If user uploads an incorrect document, then the result won’t be accurate

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 6, June 2024

DOI: 10.17148/IARJSET.2024.11649

**Data Flow Diagram (DFD)**

The first step is to draw a data flow diagram (DFD). The DFD was first developed by Larry Constantine as a way of expressing system requirements in graphical form.

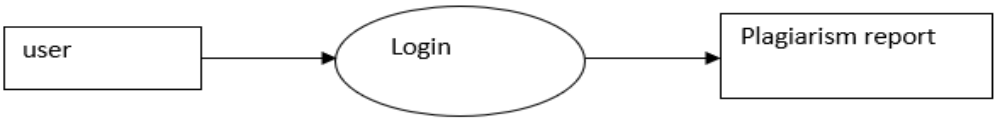
A DFD also known as a "bubble chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So, it is the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. A DFD consists of series of bubbles join by the data flows in the system.

The purpose of data flow diagrams is to provide a semantic bridge between users and systems developers. The diagrams are:

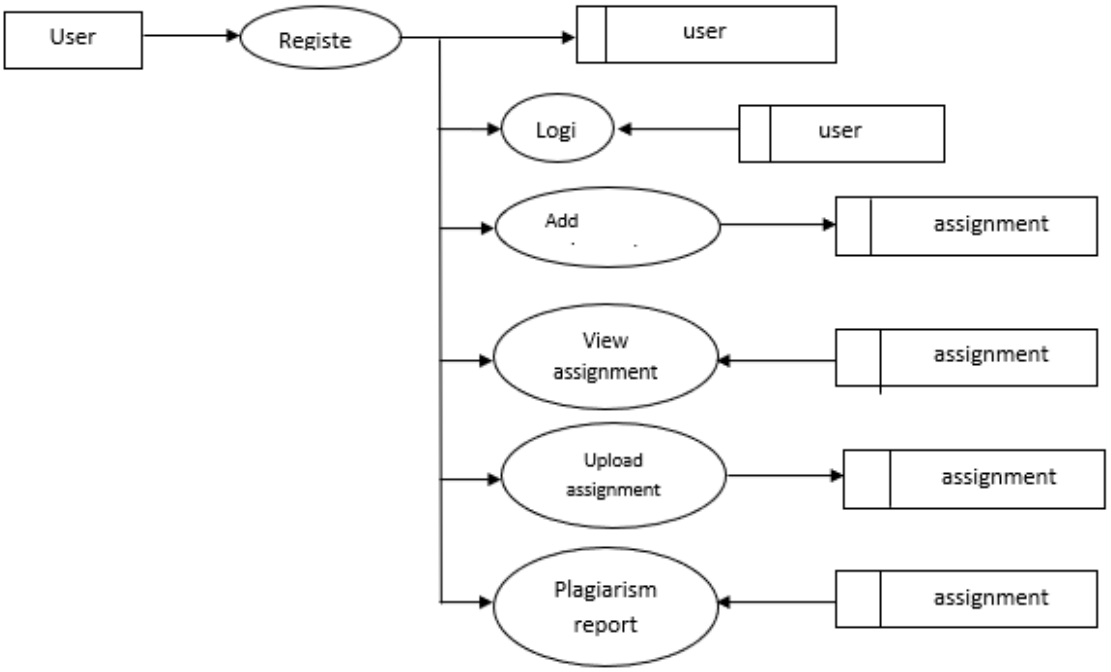
- Graphical, eliminating thousands of words;
- Logical representations, modeling WHAT a system does, rather than physical models showing HOW it does it;
- Hierarchical, showing systems at any level of detail; and
- jargon less, allowing user understanding and reviewing.

The goal of data flow diagramming is to have a commonly understood model of a system. The diagrams are the basis of structured systems analysis. Data flow diagrams are supported by other techniques of structured systems analysis such as data structure diagrams, data dictionaries, and procedure-representing techniques such as decision tables, decision trees, and structured English.

**Level 0**




**Level 1**



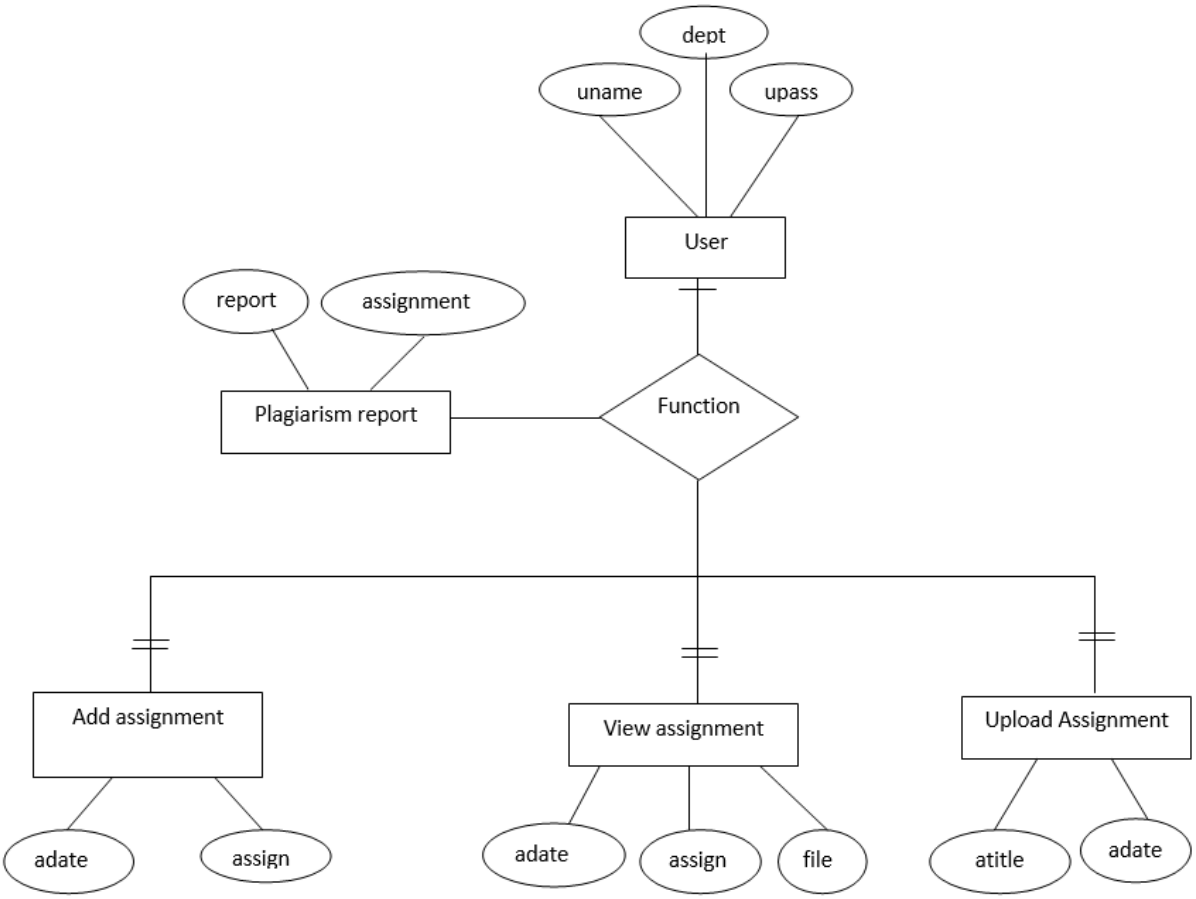
IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 6, June 2024

DOI: 10.17148/IARJSET.2024.11649

**Level 2**




The steps in proposed system are:

1) Input text: The input data or text or content is given in the text box.
2) Question type: select the question type such as objective or subjective.
3) List of sentences: the given text or content is converted into list of sentences
4) List of words: the list of sentences is again converted into list of words.
5) Stop words removal: the stop words are removed from the list of words.
6) Separation of nouns: the nouns are separated from the list of words.
7) Trivial sentences: Trivial sentences are simple, straightforward statements that convey basic information without complexity or ambiguity. These trivial sentences are sorted.
8) Extract question and answer: the extracted question and answer can be printed and downloaded as CSV, EXCEL and PDF files.

## IV. IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system and giving a user confidence in that the new system will work efficiently and effectively in the implementation stage. The stage consists of :

- Testing a developed program with sample data
- Detection and correction of error
- Creating whether the system meets a user requirement.
- Making necessary changes as desired by users.
- Training user personal

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 6, June 2024

DOI: 10.17148/IARJSET.2024.11649

The implementation phase is less creative than system design. A system design may be dropped at any time prior to implementation, although it becomes more difficult when it goes to the design phase. The final report of the implementation phase includes procedural flowcharts, record layouts, and a workable plan for implementing the candidate system design into a operational design.

**USER TRAINING**

The User Training is designed to prepare the users for testing & converting the system.
There is several ways to trail the users they are:

1) User manual
2) Help screens
3) Training demonstrations.

**1) User manual:**

The summary of important functions about the system & software can be provided as a document to the user. User training is designed to prepare the user for testing and convening a system. The summary of important functions about the system and the software can be provided as a document to the user

1. Open http page
2. Type the file name with URL index .php in the address bar
3. Index. php is opened existing user the type the username and password
4. Click the submit button

**2) Help screens:**

This features now available in every software package, especially when it is used with a menu. The user selects the "Help" option from the menu. The System success the necessary description or information for user reference.

**3) Training demonstration:**

Another user training element is a training demonstration. Live demonstration with personal contact is extremely effective for training users.

**OPERATIONAL DOCUMENTATION**

Documentation means of communication; it establishes the design and performance criteria of the project. Documentation is descriptive information that portrays the use and /or operation of the system. The user will have to enter the user name and password if it is valid he participate in auction. Otherwise if it is new user he needs to register documentation means of communication; it establishes design & performance criteria for phases of the project. Documentation is descriptive information that portrays he use &/or operation of the system.

**1) Documentation tools:**

Document production & desktop publishing tool support nearly ever aspect of software developers. Most software development organizations spend a substantial amount of time developing documents, and in many cases the documentation process itself is quite inefficient. It is not use unusual for a software development effort on documentation. For this reason, Documentation tools provide an important opportunity to improve productivity.

**2) Document restructuring:**

Creating document is far too timed consuming. If the system work's, we'll live with what we have. In some cases, this is the correct approach. It is not possible to recreate document for hundreds of computer programs.
Documentation must be updated, but we have limited resources. It may not be necessary to fully redocument an application. Rather, those portions of the system that are currently undergoing change are fully documented.
The system is business critical and must be fully redocumented. Even in this case, an intelligent approach is to pare documentation to an essential minimum.

**SYSTEM MAINTENANCE**

Maintenance is actually implementation of the review plan as important as it is programmers and analyst is to perform or identify with him or herself with the maintenance. There are psychologically personality, and professional reasons for this. Analyst and programmers spend fair more time maintaining programmer then they do writing them Maintenances account for 50-80% of total system development. Maintenance is expensive .One way to reduce the maintenance costs are through maintenance mgt and software modification audits. Thetypes of maintenance are:

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 6, June 2024

DOI: 10.17148/IARJSET.2024.11649

1. Perfective maintenance
2. Preventive maintenance

**Perfective maintenance:**

Changes made to the system to add features or to improve the performance.

**Preventive maintenance:**

Changes made to the system to avoid future problems. Any changes can be made in the future and our project can adopt the changes.

## V. RESULT ANALYSIS

This study on proposed system focuses primarily on plagiarism, which is prevalent in schools and colleges. A system can be developed for the convenience of teachers that could check the amount of plagiarism in student's assignments. This system could be mentioned as an improvement from the old manual way as it eliminates the tedious work with increased speed and efficiency.

## VI. CONCLUSION

Plagiarism detection is essential for protecting the written work. It is concluded that all institutes and teachers should be aware of plagiarism and anti-plagiarism softwares. We have designed a simple method which assists us with the detection of instances of plagiarism in assignment of school and college students. Our scheme is easy to adapt for the large variety of programming languages in use, and is sufficiently robust to be highly effective in an educational environment. While having a detection rate as good as other more complex software, it presents its report as a simple graph, enabling large numbers of assignments to be checked quickly and efficiently. By using data mining algorithm and NLP it will provides straightforward documentation which can be used as clear and convincing evidence should a suspected instance of plagiarism be disputed.

## REFERENCES

[1]. L. Prechelt and G. Malpohl, ''Finding plagiarisms among a set of programs with JPlag,'' J. Universal Comput. Sci., vol. 8, no. 11, pp. 1016–1038, Mar. 2003.
[2]. D. Grune and M. Huntjens, ''Detecting copied submissions in computer science workshops,'' Inf. FaculteitWiskunde Informatica, Vrije Universiteit, Amsterdam, The Netherlands, Tech. Rep., 1989. [Online]. Available:http://www.dickgrune.com/Programs/similarity_tester/Paper.ps
[3]. S. Schleimer, D. S. Wilkerson, and A. Aiken, ''Winnowing: Local algorithms for document fifingerprinting,'' in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2003, pp. 76–85.
[4]. Q. D. Soetens, R. Robbes, and S. Demeyer, ''Changes as fifirst-class citizens: A research perspective on modern software tooling,'' ACM Comput. Surveys, vol. 50, no. 2, pp. 1–38, Jun. 2017, doi: 10.1145/3038926.
[5]. J. Schneider, A. Bernstein, J. V. Brocke, K. Damevski, and D. C. Shepherd, ''Detecting plagiarism based on the creation process,'' IEEE Trans. Learn. Technol., vol. 11, no. 3, pp. 348–361, Jul. 2018.
[6]. K. Mierle, K. Laven, S. Roweis, and G. Wilson, ''Mining student CVS repositories for performance indicators,'' ACM SIGSOFT Softw. Eng. Notes, vol. 30, no. 4, pp. 1–5, Jul. 2005.
[7]. S. Neara, M. Vakilian, N. Chen, R. E. Johnson, and D. Dig, ''Is it dangerous to use version control histories to study source code evolution?'' in Proc. 26th Eur. Conf. Object-Oriented Program. (ECOOP), 2012, pp. 79–103
[8]. E. Giger, M. Pinzger, and H. C. Gall, ''Comparing fifine-grained source code changes and code churn for bug prediction,'' in Proc. 8th Work. Conf. Mining Softw. Repositories, 2011, pp. 83–92.
[9]. S. Negara, M. Codoban, D. Dig, and R. E. Johnson, ''Mining fifine-grained code changes to detect unknown change patterns,'' in Proc. 36th Int. Conf. Softw. Eng. (ICSE), 2014, pp. 803–813.
[10]. W. Maalej, T. Fritz, and R. Robbes, ''Collecting and processing interaction data for recommendation systems,'' in Recommendation Systems in Software Engineering. Berlin, Germany: Springer, 2014, pp. 173–197.
[11]. R. Robbes, D. Pollet, and M. Lanza, ''Replaying IDE interactions to evaluate and improve change prediction approaches,'' in Proc. 7th IEEE Work. Conf. Mining Softw. Repositories (MSR ), May 2010, pp. 161–170.
[12]. Y. Yoon and B. A. Myers, ''Capturing and analyzing low-level events from the code editor,'' in Proc. 3rd ACM SIGPLAN Workshop Eval. Usability Program. Lang. Tools, 2011, pp. 25–30.
[13]. J. Hage, P. Rademaker, and N. van Vugt, ''A comparison of plagiarism detection tools,'' Utrecht Univ., Utrecht, The Netherlands, Tech. Rep. UU-CS-2010-015, 2010.

DOI: 10.17148/IARJSET.2024.11649

[14]. M. Mozgovoy, ‘‘Enhancing computer-aided plagiarism detection,’’ Ph.D. dissertation, Univ. Joensuu, Joensuu, Kuopio, 2007.
[15]. S. Burrows, ‘‘Source code authorship attribution,’’ Ph.D. dissertation, RMIT Univ., Melbourne, VIC, Australia, 2010.
[16]. V. T. Martins, D. Fonte, P. R. Henriques, and D. da Cruz, ‘‘Plagiarism detection: A tool survey and comparison,’’ in Proc. 3rd Symp. Lang., Appl. Technol. (SLATE), vol. 38, Braganáa, Portugal, 2014, pp. 143–158.
[17]. M. Agrawal and D. K. Sharma, ‘‘A state of art on source code plagiarism detection,’’ in Proc. 2nd Int. Conf. Next Gener. Comput.Technol. (NGCT), Oct. 2016, pp. 236–241.
[18]. D. Ganguly, G. J. F. Jones, A. Ramírez-de-la-Cruz, G. Ramírez-de-la-Rosa, and E. Villatoro-Tello, ‘‘Retrieving and classifying instances of source code plagiarism,’’ Inf. Retr. J., vol. 21, no. 1, pp. 1–23, Feb. 2018.
[19]. X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker, ‘‘Shared information and program plagiarism detection,’’ IEEE Trans. Inf. Theory, vol. 50, no. 7, pp. 1545–1551, Jul. 2004.
[20]. J. A. W. Faidhi and S. K. Robinson, ‘‘An empirical approach for detecting program similarity and plagiarism within a university programming environment,’’ Comput. Edu., vol. 11, no. 1, pp. 11–19, Jan. 1987.
[21]. S. Engels, V. Lakshmanan, and M. Craig, ‘‘Plagiarism detection using feature-based neural networks,’’ ACM SIGCSE Bull., vol. 39, no. 1, pp. 34–38, Mar. 2007.
[22]. D. Gitchell and N. Tran, ‘‘Sim: A utility for detecting similarity in computer programs,’’ ACM SIGCSE Bull., vol. 31, no. 1, pp. 266–270, Mar. 1999.
[23]. M. El Bachir Menai and N. S. Al-Hassoun, ‘‘Similarity detection in java programming assignments,’’ in Proc. 5th Int. Conf. Comput. Sci. Edu., Aug. 2010, pp. 356–361.
[24]. O. Karnalim and Simon, ‘‘Syntax trees and information retrieval to improve code similarity detection,’’ in Proc. 32nd Australas. Comput. Edu. Conf., Feb. 2020, pp. 48–55.
[25]. A. Budiman and O. Karnalim, ‘‘Automated hints generation for investigating source code plagiarism and identifying the culprits on in-class individual programming assessment,’’ Computers, vol. 8, no. 1, p. 11, 2019.
[26]. P. Vamplew and J. Dermoudy, ‘‘An anti-plagiarism editor for software development courses,’’ in Proc. 7th Australas. Conf. Comput. Educ., 2005, pp. 83–90.
[27]. N. Tahaei and D. C. Noelle, ‘‘Automated plagiarism detection for computer programming exercises based on patterns of resubmission,’’ in Proc. ACM Conf. Int. Comput. Edu. Res., Aug. 2018, pp. 178–186.[34] S. Nissen, ‘‘Implementation of a fast artifificial neural network library (FANN),’’ Dept. Comput. Sci., Univ. of Copenhagen, København, Denmark, Tech. Rep., 2003.
[28]. C. Igel and M. Hüsken, ‘‘Improving the Rprop learning algorithm,’’ in Proc. 2nd Int. Symp. Neural Comput., 2000, pp. 115–121.
[29]. M. J. Wise, ‘‘Running Rabin-Karp matching and greedy string tiling,’’ Basser Dept. Comput. Sci., Sydney, NSW, Australia, Tech. Rep., 1994.
[30]. W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, ‘‘Evaluating the visualization of what a deep neural network has learned,’’ IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 11, pp. 2660–2673, Nov. 2017.