IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 4, April 2022

DOI: 10.17148/IJARCCE.2022.114158

ONLINE ASSIGNMENT PLAGIARISM CHECKER USING MACHINE LEARNING

Babitha V[1], Harshitha M[2],Hindumathi A[3], Reshma Farhin J[4]

Student, Department of Computer Science and Engineering, Adhiyamaan college of Engineering (Autonomous),Hosur, India[1]

Student, Department of Computer Science and Engineering, Adhiyamaan college of Engineering (Autonomous),Hosur, India[2]

Student, Department of Computer Science and Engineering, Adhiyamaan college of Engineering (Autonomous),Hosur, India[3]

Assistant Professor, Department of Computer Science and Engineering, Adhiyamaan college of Engineering (Autonomous),Hosur, India[4]

Abstract: Plagiarism is when someone steals someone else's idea or work and passes it off as their own. Plagiarism has been classified as a moral rights infringement in a number of countries. Plagiarism has become increasingly common in today's environment of changing technology and ever-increasing Internet usage. It is often observed in many educational areas such as research papers, blogs, articles, assignments, etc. This study focuses primarily on plagiarism, which is prevalent in schools and colleges. Many students can be found to have copied assignments from their classmates. A system can be developed for the convenience of teachers that could check the amount of plagiarism in students' 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.

Keyword: plagiarism, cosine similarity, Data mining, Hash tag, Stop word, Cleaning, stemming.

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.LITERATURE SURVEY

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

Author: Vedran Ljubovic and Enil pajic
Year: 2020
Ref link:
https://ieeexplore.ieee.org/iel7/6287639/6514899/09097285.pdf

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 code

DOI: 10.17148/IJARCCE.2022.114158

repository, and repository mining techniques are used to create "developer profiles," allowing suspect behavior to be detected. 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, Tejas Chede, Dhanashri Kuwar, Prof. Rasika Shintre
Year: 2021
Ref link:
https://www.irjet.net/archives/V8/i1/IRJET-V8I115.pdf

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.OBJECTIVE

- The main objective is to check the plagiarism report through duplicate records
- The objective is to check the plagiarism among the assignment by comparing it with the documents that are stored in database.
- To give the plagiarism report in percentage among the submitted assignment of the students.
- To show the similar sentences among the files in the submitted by the students if any presented.

IV.SCOPE

Plagiarism detection has been such an important topic in study in recent years. Plagiarism can occur in a variety of professions, including academic articles, the arts, and software code. Everything in the digitalized future will be done online; nothing will be done on paper. As a result, the plagiarism detection software will be extremely useful in the future.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 4, April 2022

DOI: 10.17148/IJARCCE.2022.114158

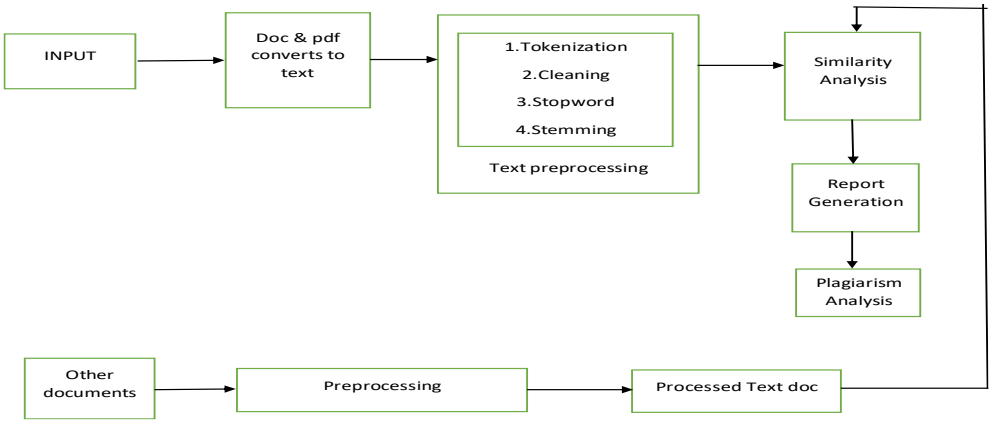
V.EXISTING SYSTEM

- 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.

VI.PROPOSED SYSTEM

A. Overview:

We will design a system to detect plagiarism in academic assignments in the suggested system, which will help to prevent students from copying other students' assignments and will improve the quality of education and also will help to improve personal skills of student and student can also check the grammar from the assignment. The plagiarism detector in this system compares comparable texts and detects plagiarism. As well semantical checking will be also done with respect to assignment. For detecting the plagiarism, we will use data mining algorithm and natural language processing.



B.Giving the input:

The input can be given as any kind of document by entering the location of the file where all the documents have been be stored all the assignments submitted by the students should be stored in that location. Once we enter the file location it will take all the document and process to the next step.

C.Conventing to text document:

Once all the files are got the step is to convert those files to text format. Whatever the file can be it can be document, word, pdf (Portable Document Format) it will be converted to text format. We are converting to text files because it will easy to do pre-processing in the text files rather than in other files.

D.Pre-Processing:

The text files will go through the pre-processing where those following process will happen and it will be ready to find the similarity in the between the files.

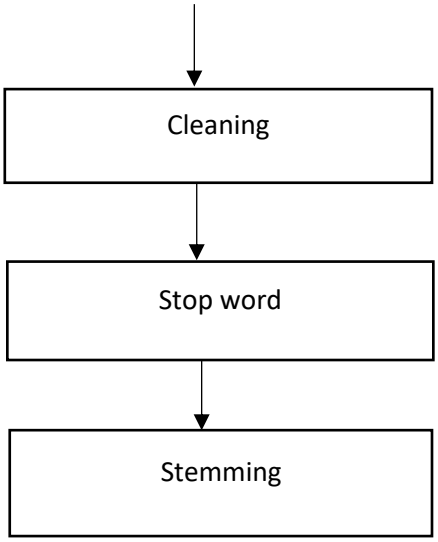
Tokenization

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 7.39 ⁜ Vol. 11, Issue 4, April 2022

DOI: 10.17148/IJARCCE.2022.114158



E.Tokenization:

The process of tokenizing or breaking a string of text into a list of tokens is known as tokenization. Tokens can be thought of as components; for example, a word in a sentence is a token, and a sentence is a token in a paragraph.

F.Cleaning:

The practise of correcting or deleting incorrect, corrupted, improperly formatted, duplicate, or incomplete data from a dataset is known as data cleaning. There are numerous ways for data to be duplicated or mislabeled when merging multiple data sources. Even if the data is right, outcomes and algorithms are untrustworthy if the data is erroneous. Because the methods differ from dataset to dataset, there is no definite way to prescribe the exact phases in the data cleaning process.

G.Stop Word:

A stop word is a widely used word (such as "the," "a," "an," or "in") that a search engine has been configured to disregard while indexing and retrieving entries as the result of a search query.

H.Steamming:

The process of developing morphological variants of a root/base word is known as stemming. Stemming algorithms or stemmers are terms used to describe stemming programmes. The phrases "chocolates," "chocolatey," and "Choco" are reduced to the root word "chocolate," and "retrieval," "retrieved," and "retrieves" are reduced to the stem "retrieve."

I.Similarity Analysis:

After removing stop words and stemming the phrases, the algorithm developed to discover similarity among text documents determines content similarity among specified texts. Various measures, such as cosine similarity, are used to quantify the similarity between document samples (assignments).

J.Report Generation:

A report is a written document that organises information for a specified audience and purpose. Although report summaries might be presented orally, entire reports are usually typically delivered in writing form. This will display the proportion of plagiarism in the files submitted by the pupil. The percentage will show to the plagiarism detected document and zero to the documents that does not any plagiarism in it.

K. Plagiarism Analysis:

Plagiarism is the act of claiming to be the author of material that someone else actually wrote. This definition relates to text documents, which is also the focus of this paper. Clearly, a question of central importance is to what extent such and similar tasks can be automated. This will show the similar lines between two documents among the submitted documents.

VII.REFERENCES:

DOI: 10.17148/IJARCCE.2022.114158

[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. Faculteit Wiskunde 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 evolu
tion?’’ 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.
[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.

DOI: 10.17148/IJARCCE.2022.114158

[28] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo, ‘‘Comparison and evaluation of clone detection tools,’’ *IEEE Trans. Softw. Eng.*, vol. 33, no. 9, pp. 577–591, Sep. 2007.
[29] C. K. Roy, J. R. Cordy, and R. Koschke, ‘‘Comparison and evaluation of code clone detection techniques and tools: A qualitative approach,’’ *Sci. Comput. Program.*, vol. 74, no. 7, pp. 470–495, May 2009.
[30] O. Karnalim, S. Budi, H. Toba, and M. Joy, ‘‘Source code plagiarism detection in academia with information retrieval: Dataset and the observation,’’ *Informat. Edu.*, vol. 18, no. 2, pp. 321–344, 2019.
[31] Z. Duric and D. Gasevic, ‘‘A source code similarity system for plagiarism”detection,’’ *Comput. J.*, vol. 56, no. 1, pp. 70–86, Jan. 2013.
[32] V. Ljubovic. (2020). *Programming Homework Dataset for Plagiarism Detection*. [Online]. Available: http://dx.doi.org/10.21227/71fw-ss32
[33] E. Pajic and V. Ljubovic, ‘‘Improving plagiarism detection using genetic algorithm,’’ in *Proc. 42nd Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2019, pp. 571–576. VOLUME 8, 2020 96513V. Ljubovic, E. Pajic: Plagiarism Detection in Computer Programming Using Feature Extraction
[34] S. Nissen, ‘‘Implementation of a fast artifificial neural network library (FANN),’’ Dept. Comput. Sci., Univ. of Copenhagen, København, Denmark, Tech. Rep., 2003.
[35] C. Igel and M. Hüsken, ‘‘Improving the Rprop learning algorithm,’’ in *Proc. 2nd Int. Symp. Neural Comput.*, 2000, pp. 115–121.
[36] M. J. Wise, ‘‘Running Rabin-Karp matching and greedy string tiling,’’ Basser Dept. Comput. Sci., Sydney, NSW, Australia, Tech. Rep., 1994.
[37] 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.