Journal of Theoretical and Applied Information Technology
30th September 2019. Vol.97. No 18

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# AN IOT BASED FRAMEWORK FOR STUDENTS' INTERACTION AND PLAGIARISM DETECTION IN PROGRAMMING ASSIGNMENTS

**HAMZA ALDABBAS**
Prince Abdullah bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University, Al-Salt- Jordan, aldabbas@bau.edu.jo

**ABSTRACT**

The Internet of Things (IoT) is the connection through the Internet of handling devices within physical objects, allowing them to move and communicate data. These devices may be used in academia to facilitate student- instructor interaction. In this research, I used IoT devices to automate an online examination system. The instructor uploaded the questions online and students provide solutions through IoT devices on University premises. The source code similarity in diverse types of source codes, however, is hard to detect because each programming language has a specific assembly of grammar. To address this issue, a code similarity detection approach was employed to extract the similarity between different source codes. The Latent Semantic Analysis (LSA) technique was used to retrieve semantic similarity by first transforming source codes into tokens to compute and then it finding semantic similarity in a pair of tokens. The dataset contained five different source codes: C, C#, C++, Python and Java.

**Keywords:** *Internet of Things, Cloud Computing, Data mining, Similarity, E-assessment*

## 1. INTRODUCTION

The Internet of things (IoT) is the connection of devices, automobiles, and home usage machines that contain microchip technology that permits objects to link, interrelate and interchange data. IoT communicates over Internet with other conventional devices, for example, desktop computers, gadgets, smartphones and tablets. These devices are embedded with expertise which can interconnect and communicate around Internet technology. These devices may be distantly supervised, observed and controlled through the use of use Radio Frequency Identifier (RFID) technology. It can be used in smart health care systems, supply and chain management, hotel management system, industry management and vehicular adhoc network [1-3] [4] [5]. The IoT network can be used in academia to automate and monitor different activities. The cupcarbon simulator is used to supervise and monitor the examination and grading of students by all connected IoT devices. The source plagiarism detection may be used to detect similar programming assignments submitted by the students. It is used to provide a quick assessment of students' programming projects. It inspires students to use their own logic and approach [6]. Source code plagiarism happens when a student uses code fragments from another source without understanding the logic. Plagiarism is a severe threat to academia in that it discourages the learning process in students. Some research suggests that every software contain code similarity in the variety of 10% to 25% [7] [8]. Different source code similarity approaches have been therefore proposed, including code plagiarism identification, bugs solutions and code clone recognition [9]. The plagiarised source code fragments may be innocent or suspicious. The source code examples given by the instructor could be used by students innocently in their assigned programming tasks. The plagiarised source codes chunks are common fragments with different in functionalities and logic. This type is

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

used as a reliable indication of plagiarism. The critical task is to detect the instances of plagiarism that can be used for further investigating the source codes [10]. Students can practice these code transformation software for programming assignments to directly translate from one source code type to other different kind [11, 12].

There are several tools developed to detect plagiarism in source codes. The tools JPlag, Sherlock, Marble, Moss, etc. need to link with other information retrieval techniques to detect plagiarism [13]. Every language parser runs the program on its parse tree. The parse tree is also called syntax tree which works on syntax rules of that specific language. This information can be used to analyze programming language behaviour [14]. But when more than two languages are involved then it is difficult to detect plagiarism in these multiprogramming languages because of mixed programming languages grammar.

In this paper, IoT network in Al-Balqa Applied University Jordon was used to automate the examination system through students' evaluations. There are fixed sensors installed in the premises of the university and students use IoT devices to communicate with these sensors. The instructor gives programming assignment question online, and then students upload the solutions through IoT devices. Thus students are not bound to submit their assignments physically to the instructor. They can submit their assigned tasks from anywhere. A methodology is here proposed to find plagiarism in source code assignments where students copy code from someone else or from the internet. The LSA is an information retrieval technique used to extract similarity values based on semantics. It is independent of the specific programming language's rules. To investigate the research, a dataset is acquired in the various source codes.

## 2. LITERATURE REVIEW

The Blackbox technique is useful to rank similar statements and functionalities in source code. In [15] the authors used this technique to extract the same code sequence in students' source code tasks. To investigate the similarity, they used the coding style metric of the user that reflects the personality of the programmer. The similarity issue does not arise due to students' inability only, but bad time management is also a big factor. In [16], the author used the low-level instruction approach to measure the source code similarity instead of tokens based comparison. The Java byte code approach is applied to distinguish the plagiarism attacks in source codes. Many of the students recycled text descriptions methods for similarity identification. But the hybrid method is useful for these both methods. In [17], the hybrid method is used to excerpt the resemblance descriptions from transitional code creation. Additional, the classification method is programmed to identify the same code segments. In [18], the authors used LSA to predict similarity in students' codes. This technique is used collectively with PlaGate to examine similarity among programming languages. More, it is explained how different code portions are important as far as the similarity is concerned. The parse tree can use the parsing of code based on source code from the diverse associate sources. In [19], the parsing tree kernel technique is used to extract similar text. This procedure does not postulate a better effect due to unbalanced differences in code functionality.

Additionally, the algorithm used for learning purposes is projected for enhancing the similarity correctness. The data extraction is a dynamic research area, which still has the deficiency to extract plagiarized functions from a huge corpus. In [20], A technique of substring assessment in codes is used to retrieve similarity. Further, the p-values technique is used for text similarity. In [21], the recognition procedure constructed using the parse tree to authenticate duplicates among C documents. This source code clone abstraction used three steps elementary, classification and generalisation. The proposed idea is used on collected data from students' source codes' tasks. The differentiating of similar code replicas in source code is important to measure of computer science

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

domain. The reprocessing of source code is a rising question in software engineering life cycle. The active characterization of executable code is used to identify the similar text [22]. The similar source code fragments are distinguished in source code by calculating the similarity between functionalities of different statements. Diverse algorithms are designed to extract information for similarity in the code. In [23], An enhanced Combined Method (CM) algorithm is applied to extract plagiarism among scholars' source codes. Plagiarism detection in codes is a common issue for programming assignments especially in the students' projects related to source code writing. In [24], the authors described a real-time technique to notice the similarity in students' source codes to progress the learning growth in students. The developed tool is used to catch the plagiarised text in C programming assignments' functionalities. The summary of the text documents can be obtained using the fingerprint process for the text documents. the IoT network is used to automate the e-assessment process in terms of source code similarity. Previously mostly the authors detected plagiarism in single or between two different programming languages and without the IoT technology but the proposed research detects plagiarism among C, C++, Java, Python and C# source codes using IoT devices to automate, fast and efficient process.

## 3. PROPOSED METHODOLOGY: IOT BASED STUDENTS' INSTRUCTOR INTERACTION MODEL WITH SOURCE CODE SIMILARITY AMONG PROGRAMMING LANGUAGES

This University' LMS provides students' instructor communication based on e-learning as follows. The teacher gives programming tasks to students online and in response students submit the solutions.

It offers the virtual classrooms ability to students based on interactive lectures [25] [26]. The source code similarity in students' programming assignments between different programming languages is a big challenge. The IoT network may be used to automate the system regarding the type of hardware, time and distance. The sensors are installed in the premises of the university and students use IoT devices to communicate using these sensors. Further, software plagiarism method is proposed to calculate source code similarity between students' programming assignments. To test the proposed approach. I took two case studies (stack and binary search) in five unique source codes, i.e. C, C++ C#, Python and Java. Teacher gives programming assignment to students in any of these five languages, and the proposed methodology detects plagiarism in the source code based on semantics. The LSA is used to retrieve similarity between a pair of documents written in different languages. It uses a mathematical algorithm called SVD used to extract text summary or essay grading [27] [28]. First, the source code is preprocessed and change it to the document-term matrix. The preprocessing includes, stemming, root words, minimum and maximum frequencies of each term and removing noise. Further, the different types of weighting filters are used to categorize tokens according to the similarity contributions [18]. The weighted results rank the terms regarding their participation in similarity.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

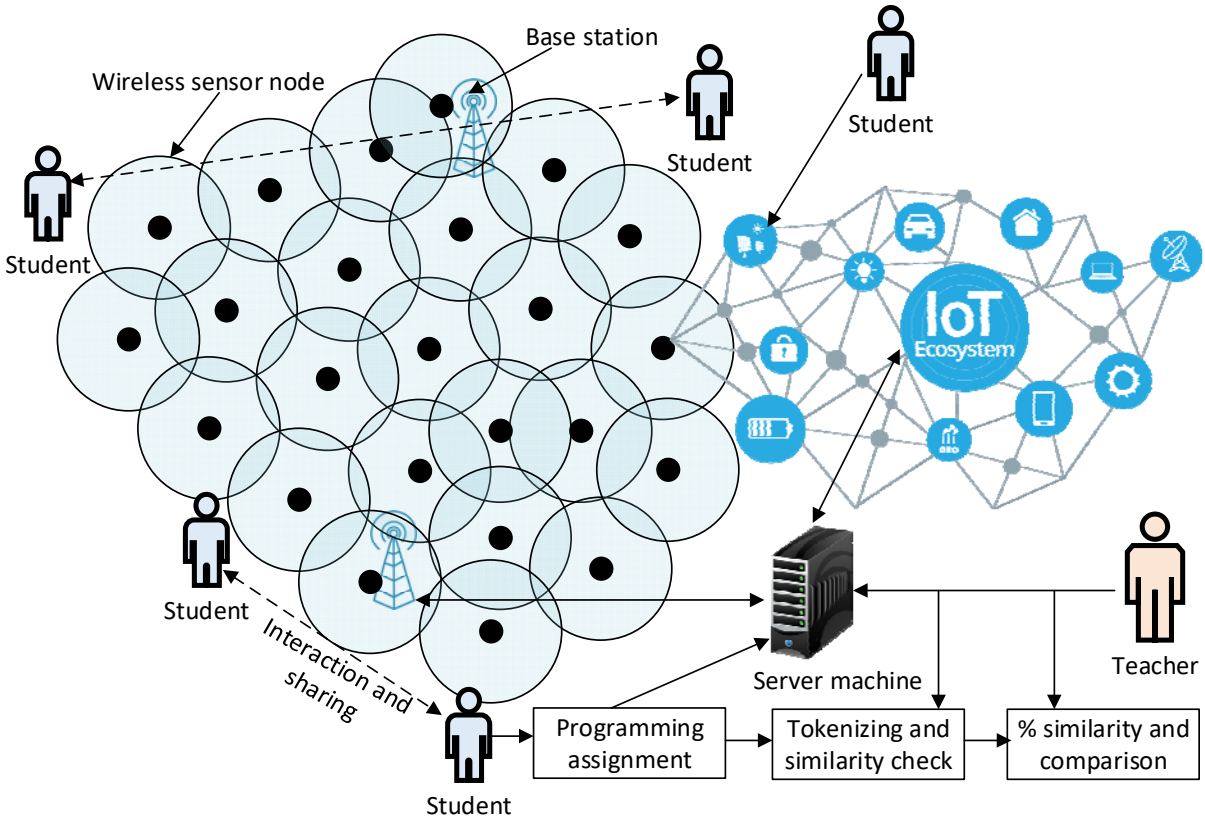


*Figure 1: IoT based students' interaction and Plagiarism Detection in programming assignments*

The range of +1 and -1 is extracted using similarity among the files if I need to calculate the semantic similarity using the cosine similarity measure, which is used by LSA. To range the resultant semantic values in 0 to 2 scale the normalization method is applied. The LSA technique does not affect the syntax and grammatical rules of any programming language because it uses a bag of words model. It takes terms as a bag of words and then extracts the semantic values based on the contribution of the similarity of each term. Every language has a specific structure and grammar rule, but still, LSA detects plagiarism between a pair of two different programming languages [29, 30]. The percentage similarity value is calculated between each pair of varying source code documents. It gives the overall contribution of each normalized token which shows better results to the instructor.

The proposed approach works using following Algorithm 1.

1) *Decompose the source codes S into a set of tokens set T such that T ∈ S, set k=1*
2) *Construct a term frequency Matrix M from T*
3) *Perform Entropy weighting to zoom the importance of each token*
4) *Perform SVD on matrix M to extract singular value matrix* $V^T = (v_{i1}, v_{i2},..v_{ir})$
5) *Compute Similarity among Pairs of SVD vectors.*
6) *Notify Similarity*

First, we preprocess source code corpus to convert it to tokens without noisy data. Then, term frequencies are computed for each token to show the occurrences of each token. After that, entropy weighting technique is applied to zoom the importance of each token. Then, the SVD technique is applied to reduce the dimensions of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

source codes data without losing actual information. Finally, LSA algorithm is applied to compute similarity among each pair of tokens.

## 4. RESULTS AND DISCUSSIONS

Students are not limited to sit in the library or the classroom and to submit any given tasks physically. They may read and update about any activity regarding their studies through their IoT devices. Instructor can give any task to students by uploading online to the system. Then, students can upload the solutions of the tasks from anywhere within the range of these sensors. The Al Balqa Applied University sensors network is being simulated in figure 1 and figure 2. The online examination system has the feature to analyze the similarity between different source codes solved by the students. I used cupcarbon simulator to extract students' and instructor information from IoT devices. Sensors are configured in different regions of Al-Balqa Applied University Jordon using cupcarbon simulator.

Different programming languages' code can be used to calculate the plagiarism among them using semantic similarity. In the current study, we have used five programming languages for the experiments. These programming languages are C, C#, C++, Python and Java. Semantic similarity between a pair of different source codes in the same case study using the LSA technique. It extracts semantic similarity between tokens and does not use the grammar rule of any specific programming language. There are twenty different semantic similarity tables are extracted from five programming languages. The preprocessing method is applied to remove the noisy words, characters and symbols and to get to tokens that can be used for plagiarism detection analysis. The LSA technique process theses tokens for semantic analysis. To retrieve information from source code first it needs to be converted to tokens' weighting information. It contains the extracted tokens with weighting values that show the contribution of each symbol.

Sensors are deployed in, and every student in the university is using the smartphone. Through this, I may be notify by the location of every student, their activities, etc. I have deployed three base stations in the area of the university with a number of sensors as shown in figure 2 and sensors are sending the data to the base station as shown in figure 3. The base station is coloured with yellow because it's inactive state and sensors send the data keeping the delay of 1 minute or as you want to receive the data. This system is deployed in the Al-Balqa Applied University to maintain a check and balance on every student and further to update the activities of every student.

Journal of Theoretical and Applied Information Technology
30th September 2019. Vol.97. No 18

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

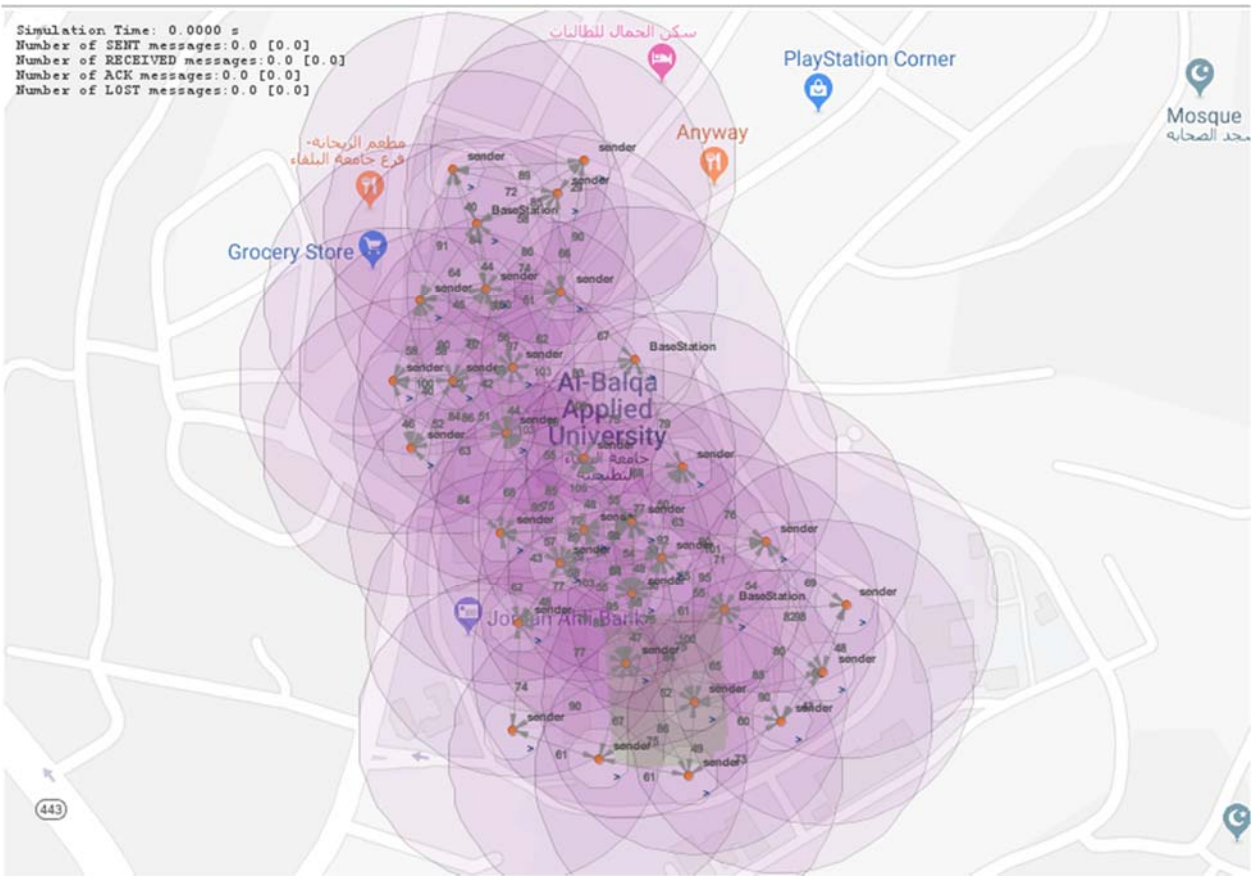


*Figure 2: Before the simulation*

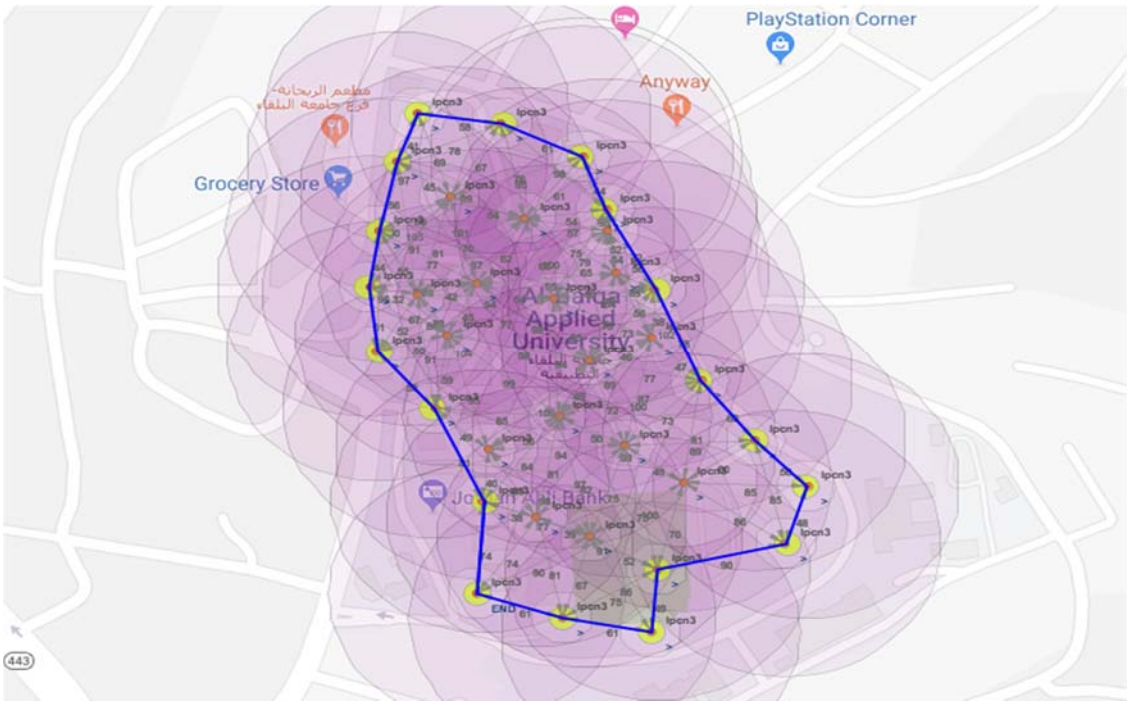


*Figure 3: Finding the boundary of Al-Balqa Applied University*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

I have deployed the sensors in Al-Balqa Applied University, and it's recognizing the edge of the university as shown in figure 4. When the students enter in the university, these sensors send the data to the base station that student have entered in the campus premises and whenever the student leaves the university it will also tell that the student is out of the boundary of the university. If an unknown person is entering the edge of the university, it sends the data against this activity.

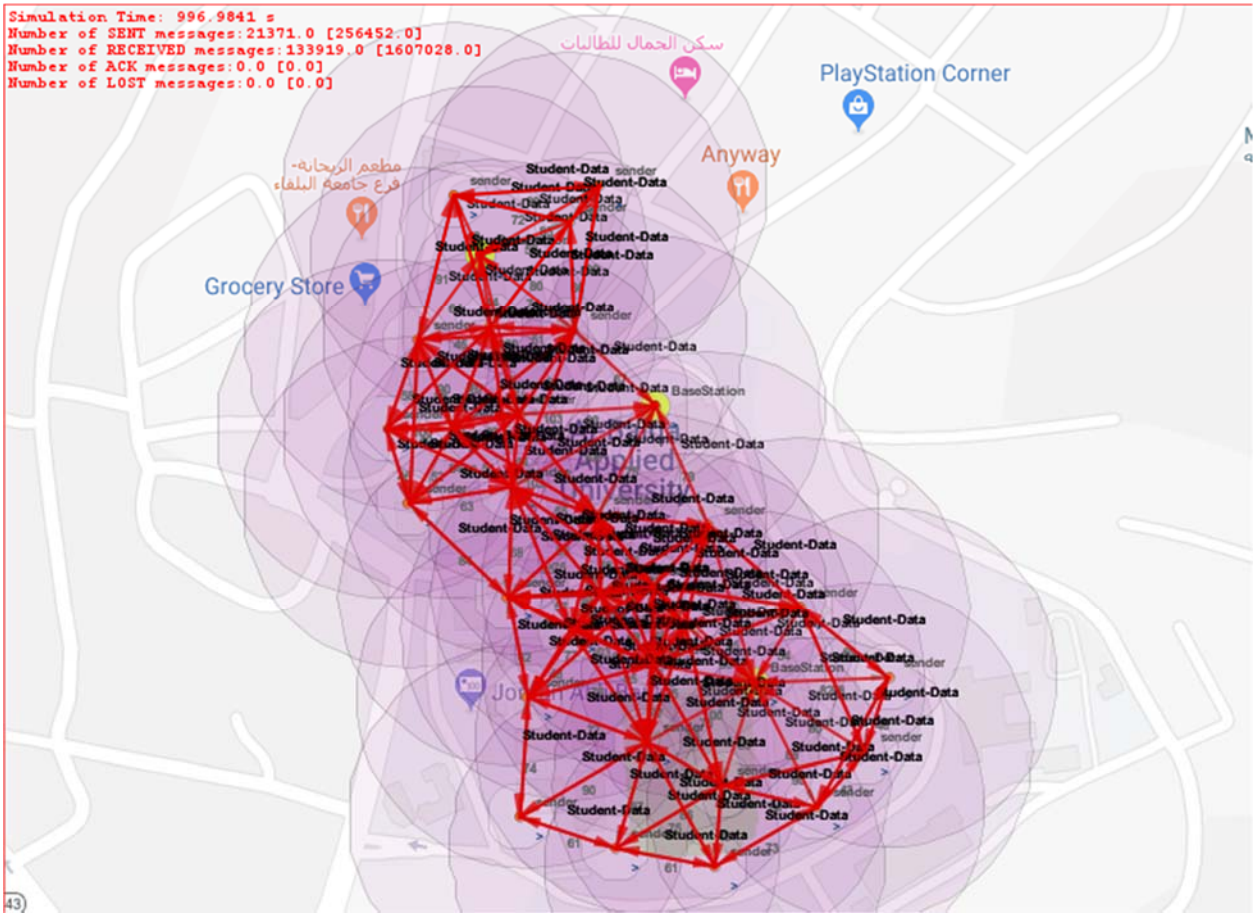


*Figure 4: Sensors sending the data to the base station*

The energy consumption of the nodes is shown in figure 5a and figure 5b, which is in Joules. The remaining energy of the node and the use of the battery can be taken as a time function. The figure 5a shows that the node is consuming 0.06 Joule, and this is specially done when the node sends a message to other nodes. The energy consumption for its electric operation and data propagation, i.e. both for broadcasting and receiving, are the energy-hungry processes. So, a time function can be considered for the decreasing of energy if i take the remaining prior into account. In my simulation, i have done a simulation for 996 seconds and the figures 5a,5b are showing the energy consumed by sensors in 996 seconds.

Journal of Theoretical and Applied Information Technology
30th September 2019. Vol.97. No 18

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

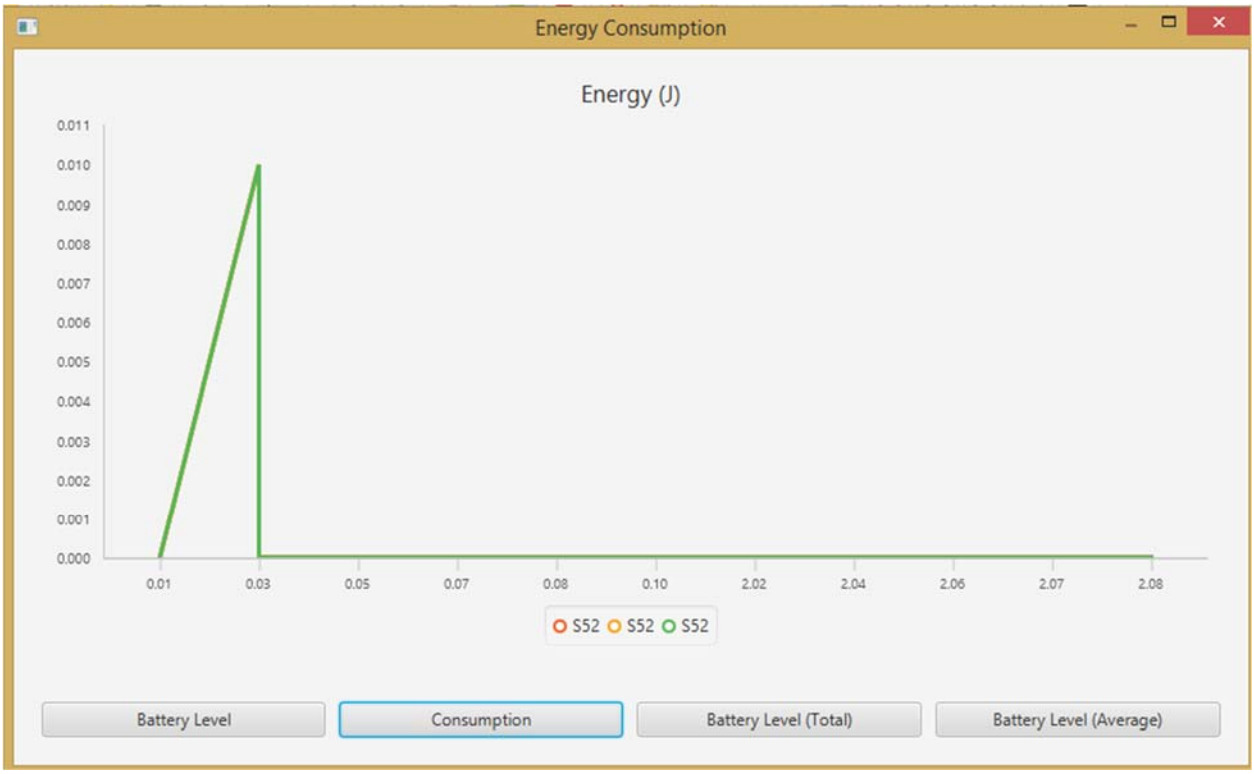


*Fig 5 a: Energy consumption of sensor 1 as a function of time*


*Fig 5 b: Remaining energy of sensor 1 as a function of time*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1 shows the source code similarity calculations among C++ and C tokens. The tokens are shown in the first column, which are extracted from both source codes, the last two columns show the semantic similarity values for C and C++ respectively. The values which are close to each other are more useful for plagiarism detection. For example, array, bottom, value, while, find etc. This represents that these tokens have close weighting values. The last few tokens binary search, cin, class, cout, invalid, result and void show distinct similarity values because these tokens are present in C but not in C++. The semantic similarity is calculated for a case study related to stack and the programming languages used are C and C++ as shown in Table 1. The tokens show in the first column, the semantic similarity values for Java and Python are shown in the last two columns as shown in Table 2. Moreover, these programming languages are different at syntax and semantic level. I did simulation in R language for information retrieval from the text.

*Table 1: Similarity values between C & C++*

| Tokens | C | C++ |
|---|---|---|
| array | 1.12446842 | 1.30487059 |
| bottom | 1.19040993 | 1.10192889 |
| break | 1.05360563 | 1.13130061 |
| element | 1.07640164 | 1.0301734 |
| else | 1.14167912 | 1.03309946 |
| enter | 1.16877055 | 1.04892579 |
| find | 1.07640164 | 1.0301734 |
| first | 1.05360563 | 1.13130061 |
| for | 1.07640164 | 1.0301734 |
| found | 1.08496291 | 1.20810655 |
| if | 1.15280329 | 1.0603468 |
| int | 1.15764637 | 1.02167845 |
| integer | 1.07640164 | 1.0301734 |
| last | 1.08496291 | 1.20810655 |
| list | 1.05360563 | 1.13130061 |
| locate | 1.07640164 | 1.0301734 |
| main | 1.07640164 | 1.0301734 |
| middle | 1.16992583 | 1.4162131 |
| number | 1.15501393 | 0.97394383 |
| present | 1.05360563 | 1.13130061 |
| printf | 1.13856855 | 1.33940717 |
| return | 1.1054151 | 1.00942074 |
| scanf | 1.10721126 | 1.26260123 |
| search | 1.24682878 | 1.06157378 |
| top | 1.18444919 | 1.08732876 |
| value | 1.07640164 | 1.0301734 |
| while | 1.07640164 | 1.0301734 |
| ascend | 1.09919766 | 0.92904619 |

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

| | | |
|---|---|---|
| binarysearch | 1.19839532 | 0.85809237 |
| cin | 1.19839532 | 0.85809237 |
| class | 1.09919766 | 0.92904619 |
| cout | 1.25642223 | 0.81658705 |
| invalid | 1.09919766 | 0.92904619 |
| positive | 1.09919766 | 0.92904619 |
| public | 1.09919766 | 0.92904619 |
| result | 1.23032983 | 0.83525035 |
| searchfun | 1.23032983 | 0.83525035 |
| void | 1.29759298 | 0.78713856 |

*Table 2: Similarity values between Java & Python*

| Tokens | Java | Python |
|---|---|---|
| arg | 1.07509626 | 1.106359846 |
| catch | 1.07509626 | 1.106359846 |
| class | 1.07509626 | 1.106359846 |
| empty | 1.08254551 | 1.008803854 |
| emptystackexception | 1.07509626 | 1.106359846 |
| int | 1.07509626 | 1.106359846 |
| integer | 1.15019252 | 1.212719692 |
| main | 1.07509626 | 1.106359846 |
| new | 1.11902476 | 1.168576367 |
| out | 1.21082186 | 1.298589837 |
| pop | 1.23173453 | 1.024715544 |
| print | 1.19166473 | 1.020441917 |
| println | 1.11902476 | 1.168576367 |
| public | 1.11902476 | 1.168576367 |
| push | 1.22172746 | 1.034584445 |
| stack | 1.20901951 | 1.07982423 |
| stack1 | 1.27651613 | 1.088140317 |
| static | 1.15019252 | 1.212719692 |
| string | 1.07509626 | 1.106359846 |
| system | 1.21082186 | 1.298589837 |
| test | 1.07509626 | 1.106359846 |
| void | 1.15019252 | 1.212719692 |
| append | 1.08999475 | 0.911247863 |
| contain | 1.08999475 | 0.911247863 |
| current | 1.08999475 | 0.911247863 |
| def | 1.17998951 | 0.822495725 |
| display | 1.20896134 | 0.793923919 |

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

| | | |
|---|---|---|
| displaystack | 1.08999475 | 0.911247863 |
| else | 1.14263831 | 0.85933119 |
| for | 1.08999475 | 0.911247863 |
| full | 1.08999475 | 0.911247863 |
| if | 1.14263831 | 0.85933119 |
| input | 1.17998951 | 0.822495725 |
| item | 1.14263831 | 0.85933119 |
| key | 1.17998951 | 0.822495725 |
| len | 1.14263831 | 0.85933119 |
| press | 1.17998951 | 0.822495725 |
| read | 1.17998951 | 0.822495725 |
| stacksize | 1.14263831 | 0.85933119 |
| value | 1.14263831 | 0.85933119 |
| when | 1.17998951 | 0.822495725 |

Figure 6 shows the tokens-based source code similarity between Java and C for the binary search case study to examine semantic similarity values. The horizontal line shows the tokens extracted from C and Java and the vertical line indicates the similarity values of each symbol. The blue colour represents tokens of C, and the orange colour represents tokens of Java. The similarity values behave in the range of 0 to 1.4. The println, out, new, system and indexint tokens have distinct values because these present in one source code but not in other. These types of symbols still show semantic similarity values based on LSA. The bottom, print and top tokens in Java show the highest value which is 1.4 while the system, out and println symbols show lowest similarity values in java. The system and middle tokens show the highest value in C which is 1.3. The overall percent similarity between these source codes is 78.235%. Figure 7 shows the source code similarity between python and C++ tokens. The calculated similarity is shown y-axis while symbols are showed on the x-axis. The blue and orange colour shows tokens of C++ and Python respectively. The arr token shows the highest value in Python which is 1.5. The find, ascend, for, and integer show improved similarity values between both source codes because their costs are close to each other. These types of tokens more useful for semantic similarity to detect plagiarism. The overall semantic similarity between C++ and Python is 70.647% in the binary search case study. These types of tokens are beneficial for plagiarism detection. The token-based comparison between C# and C++ is stack case study is shown in Figure 9. The tokens similarity is analyzed vertically while the tokens details are given horizontally.

Journal of Theoretical and Applied Information Technology
30th September 2019. Vol.97. No 18

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

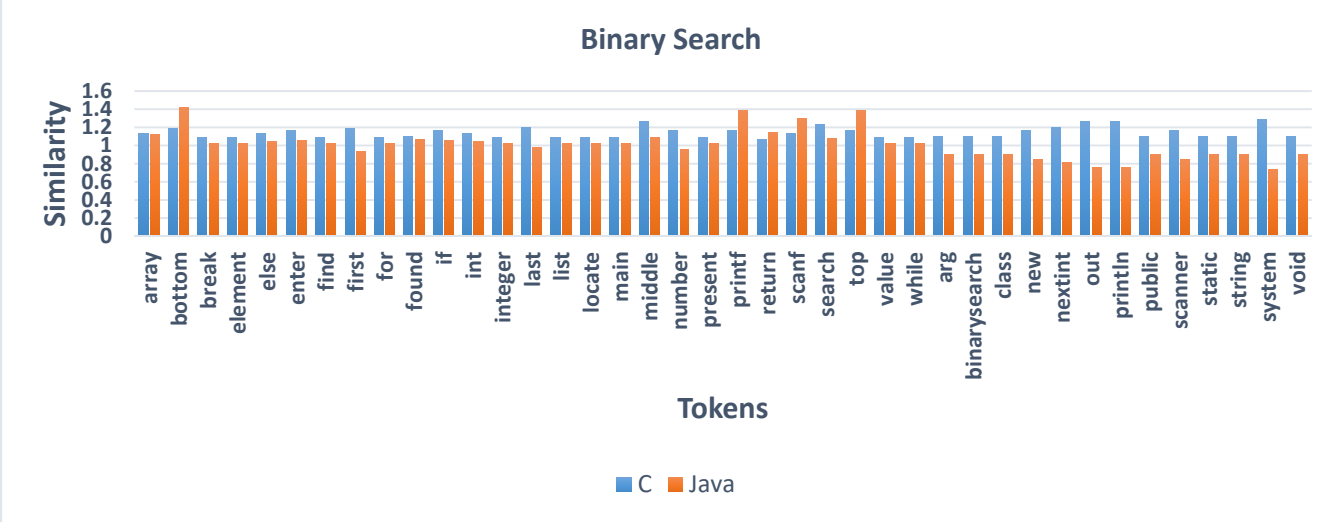


*Figure 6: C and Java tokens similarities*

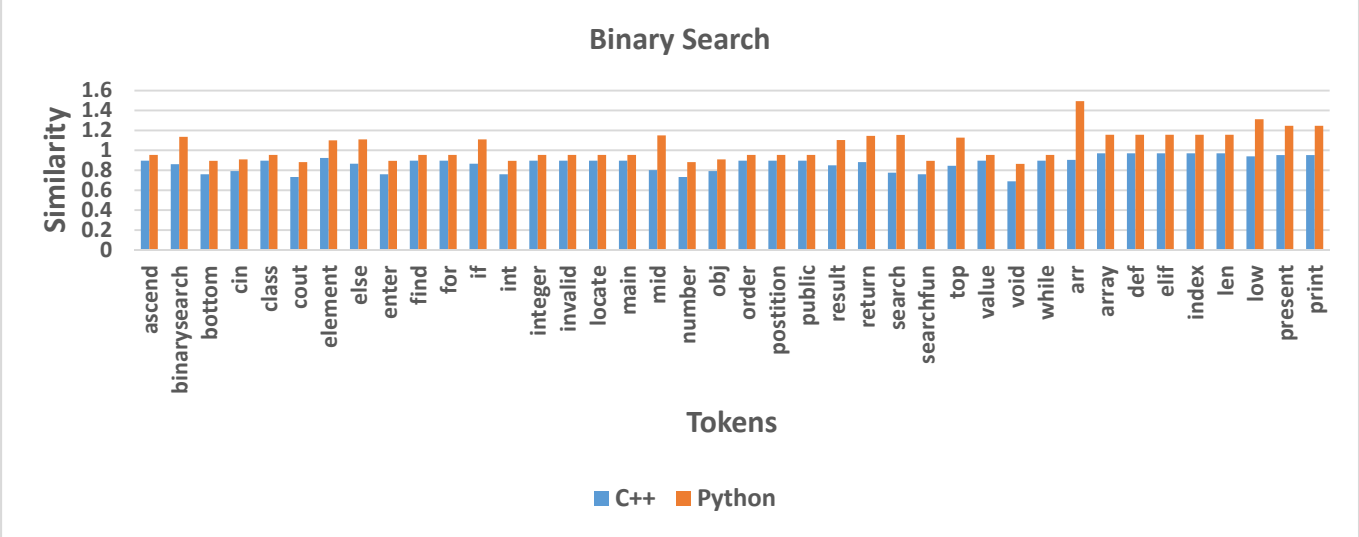


*Figure 7: C++ and Python tokens' similarities*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

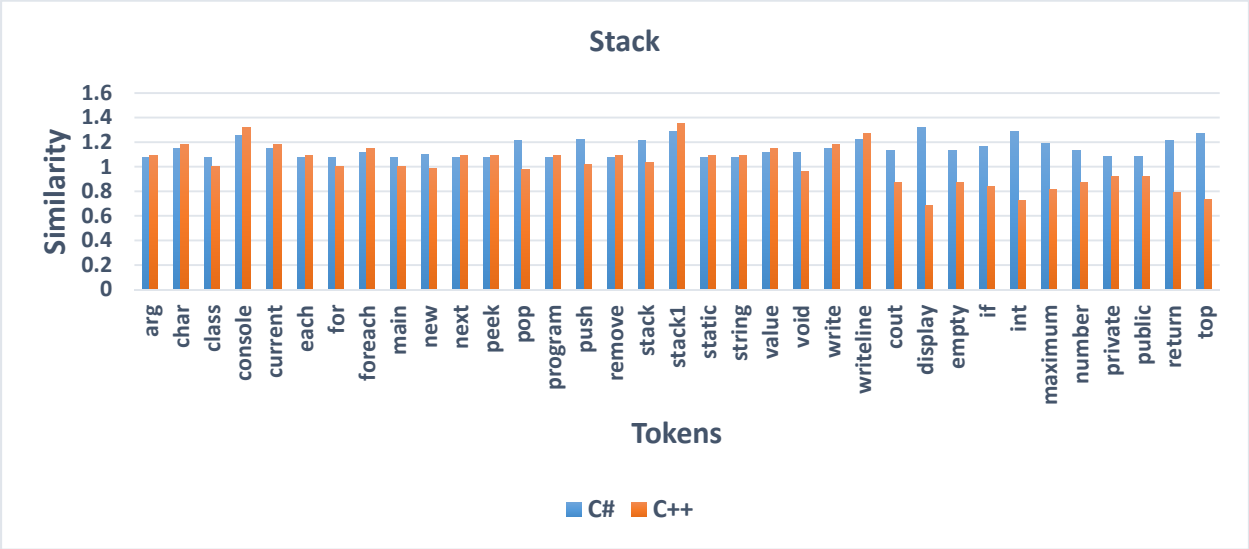


*Figure 8: C# and C++ tokens' similarities*

## 5. CONCLUSION AND FUTURE DIRECTIONS

The IoT is an emerging technology that may be used in academia for teaching and examination activities. In this paper, an IoT model is proposed to connect students and instructor for academic purposes. Instructors and students are connected with university's online database through IoT devices. The cupcarbon simulator is used to manager and analyze the activities of the connected devices. It can record the entering and going time and other academic activities of each connected device in the university premises. Further, a methodology is proposed to catch the similarity in students' source codes. The students' capability for learning programming can be enhanced and made more effective when students' programming assignments' plagiarism detection is done. The LMS is used to interact with students with instructors. The students get training from available courses and then resolve the given programming assignments. A proper plagiarism detection method is required to extract similarity in students' source code. The source code similarity among different codes is a serious issue. A new assessment methodology is used to extract similar code fragments among several source codes. The LSA is the Natural Language Processing technique which is used to extract semantic similarity between source codes. These source codes are used in two (binary search, stack) different programming problems. The pre-processing steps are performed on five different source codes to extract meaningful tokens. The preprocessing steps contain stemming, root words, tokens contribution and frequencies. The weighting of the tokens followed the process. The next step is SVD, which is performed to extract the most relevant tokens from source codes. This research idea may be used in the education system for obtaining resemblances in students' source codes. In future, I will try to embed some plagiarism detection tool to analyze programming languages'' fragments for semantic similarity. Also, the deep and machine learning algorithms may also be used to extract features and analyses the comparison with LSA technique.

## REFERENCES

[1] Habte, T.T., et al., *IoT for Healthcare*, in *Ultra Low Power ECG Processing System for IoT Devices*. 2019, Springer. p. 7-12.
[2] Bagheri, M. and S. Haghighi Movahed. *The effect of the Internet of Things (IoT) on education business model.* in *2016 12th*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

*International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. 2016. IEEE Computer Society.

[3] Luthra, S., et al., *Internet of Things (IoT) in Agriculture Supply Chain Management: A Developing Country Perspective*, in *Emerging Markets from a Multidisciplinary Perspective*. 2018, Springer. p. 209-220.

[4] Gnanamalar, R., et al., *IoT Driven Vehicle License Plate Extraction Approach.* International Journal of Engineering & Technology, 2018. **7**(2.24): p. 457-459.

[5] Kotha, H.D. and V.M. Gupta, *IoT Application, A Survey.* International Journal of Engineering & Technology, 2018. **7**(2.7): p. 891-896.

[6] Bradley, S. *Managing plagiarism in programming assignments with blended assessment and randomisation*. in *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. 2016. ACM.

[7] ShanmughaSundaram, M. and S. Subramani, *A Measurement of Similarity to Identify Identical Code Clones.* International Arab Journal of Information Technology (IAJIT), 2015. **12**.

[8] Roy, C.K. and J.R. Cordy, *A survey on software clone detection research.* Queen's School of Computing TR, 2007. **541**(115): p. 64-68.

[9] Ragkhitwetsagul, C. *Measuring Code Similarity in Large-scaled Code Corpora*. in *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*. 2016. IEEE.

[10] Joy, M., et al., *Source code plagiarism—a student perspective.* IEEE Transactions on Education, 2011. **54**(1): p. 125-132.

[11] Buddrus, F. and J. Schödel. *Cappuccino—A C++ to Java translator*. in *Proceedings of the 1998 ACM symposium on Applied Computing*. 1998. ACM.

[12] Ullah, F., et al., *Plagiarism detection in students' programming assignments based on semantics: multimedia e-learning based smart assessment methodology.* Multimedia Tools and Applications, 2018: p. 1-18.

[13] Heres, D. and J. Hage. *A Quantitative Comparison of Program Plagiarism Detection Tools*. in *Proceedings of the 6th Computer Science Education Research Conference*. 2017. ACM.

[14] Mou, L., et al. *Convolutional Neural Networks over Tree Structures for Programming Language Processing*. in *AAAI*. 2016.

[15] Mirza, O.M., M. Joy, and G. Cosma. *Suitability of BlackBox dataset for style analysis in detection of source code plagiarism*. in *Innovative Computing Technology (INTECH), 2017 Seventh International Conference on*. 2017. IEEE.

[16] Karnalim, O. *Detecting Source code plagiarism on introductory programming course assignments using a bytecode approach*. in *Information & Communication Technology and Systems (ICTS), 2016 International Conference on*. 2016. IEEE.

[17] Yasaswi, J., et al. *Unsupervised learning based approach for plagiarism detection in programming assignments*. in *Proceedings of the 10th Innovations in Software Engineering Conference*. 2017. ACM.

[18] Cosma, G. and M. Joy, *An approach to source-code plagiarism detection and investigation using latent semantic analysis.* IEEE transactions on computers, 2012. **61**(3): p. 379-394.

[19] Son, J.-W., et al., *An application for plagiarized source code detection based on a parse tree kernel.* Engineering Applications of Artificial Intelligence, 2013. **26**(8): p. 1911-1918.

[20] Bakker, T., *Plagiarism Detection in Source Code*. 2014, Ph. D. dissertation, Universiteit Leiden.

[21] Lazar, F.-M. and O. Banias. *Clone detection algorithm based on the Abstract Syntax Tree approach*. in *Applied Computational Intelligence and Informatics (SACI), 2014 IEEE 9th International Symposium on*. 2014. IEEE.

[22] Jhi, Y.-C., et al., *Program characterization using runtime values and its application to software plagiarism detection.* IEEE Transactions on Software Engineering, 2015. **41**(9): p. 925-943.

[23] Ohno, A. and H. Murao, *A two-step in-class source code plagiarism detection method utilizing improved CM algorithm and SIM.* International Journal of Innovative Computing, Information, and Control, 2011. **7**(8).

[24] Pawelczak, D. *Online detection of source-code plagiarism in undergraduate programming courses*. in *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*. 2013. The Steering

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[25] Zhang, D., *Interactive multimedia-based e-learning: A study of effectiveness.* The American Journal of Distance Education, 2005. **19**(3): p. 149-162.

[26] Zhang, D., et al., *Can e-learning replace classroom learning?* Communications of the ACM, 2004. **47**(5): p. 75-79.

[27] Zhiyuan, Z., *Latent Semantic Analysis.* 2017.

[28] Ullah, F., et al., *Software plagiarism detection in multiprogramming languages using machine learning approach.* Concurrency and Computation: Practice and Experience: p. e5000.

[29] Hájek, P., *Combining bag-of-words and sentiment features of annual reports to predict abnormal stock returns.* Neural Computing and Applications, 2017: p. 1-16.

[30] Ullah, F., *An E-Assessment Methodology Based on Artificial Intelligence Techniques to Determine Students' Language Quality and Programming Assignments' Plagiarism.* Intelligent Automation & Soft Computing, 2019: p. 12.