AcPgChecker: Detection of Plagiarism among Academic and Scientific Writings

Preprint · July 2021

CITATIONS
0

READS
169

6 authors, including:

Atish Kumar Dipongkor
University of Dhaka
15 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Rayhanul Islam
University of Dhaka
9 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Md Shafiuzzaman
Jashore University of Science and Technology
9 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Asif Nashiry
Universite des Sciences et Techniques de Masuku
22 PUBLICATIONS **134** CITATIONS
SEE PROFILE

2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision Pattern Recognition (icIVPR)

AcPgChecker: Detection of Plagiarism among Academic and Scientific Writings

Atish Kumar Dipongkor*, Rayhanul Islam†, Md. Shafiuzzaman*, Md Asif Nashiry*,
Syed Md. Galib*, Khaza Moinuddin Mazumder‡,
*Jashore University of Science and Technology, Bangladesh
Email: {atish.cse,md.shafiuzzaman.cse,asif.nashiry,galib.cse}@just.edu.bd
†University of Dhaka, Bangladesh
Emails: rayhanul.islam@du.ac.bd
‡University of Dhaka, Bangladesh
Emails: moinmazumderiitdu@gmail.com

***Abstract*—The unacknowledged usage of Intellectual Property (IP), for example, academic or scientific writings, is considered plagiarism. People get involved in this unethical practice in order to achieve rewards or society's attention effortlessly. For example, students often copy & paste from other's documents to achieve better marks. However, it is crucial to detect plagiarism to reward the actual owners of IP like academic or scientific writings. The existing studies are better at identifying external online sources of plagiarism for a given document. However, busy academics need to identify plagiarism among a set of offline writings or documents more often. To this end, we develop a tool named AcPgChecker to assist busy academicians. Initially, this tool measure similarity among a set of documents using a well-known information retrieval technique, Cosine Similarity. Then, it compares the measured similarities with a predefined threshold to detect plagiarism. The main attractions of this tool are: it is open source and freely available for anyone whereas equivalent existing tools are very expensive.**

Contribution: Offline plagiarism detection among academic writing and scientific document.

***Index Terms*—Plagiarism, Cosine Similarity, Academic Writing, Scientific Writing, Intellectual Property**

Assignment of Student X
Colonialism had a destabilizing effect on what had been a number of ethnic groups that is still being felt in African politics. Before European influence, national borders were not much of a concern, with Africans generally following the practice of other areas of the world, such as the Arabian peninsula, where a group's territory was congruent with its military or trade influence.** In the 1870s European nations were bickering over themselves about the spoils of Africa. In order to prevent further conflict between them, they convened at the Berlin Conference of 1884-1885 to lay down the rules on how they would partition up Africa between themselves. Between 1870 and World War I alone, the European scramble for Africa resulted in the adding of around one-fifth of the land area of the globe to its overseas colonial possessions. *** ** https://bit.ly/3cZzgMJ *** https://bit.ly/3t3XqeA
Assignment of Student Y
Colonialism had a destabilizing effect on what had been a number of ethnic groups that is still being felt in African politics. In the 1870s European nations were bickering over themselves about the spoils of Africa. Between 1870 and World War I alone, the European scramble for Africa resulted in the adding of around one-fifth of the land area of the globe to its overseas colonial possessions. Prior to European influence, national borders were not much of a concern, with Africans generally following the practice of other areas of the world, such as the Arabian peninsula, where a group's territory was congruent with its military or trade influence.

Fig. 1: Example of plagiarism in students' assignments

I. Introduction

Nowadays, people often claim other people's ideas, statements, or works as their own in articles, reports, or other writings. Students are also not free from this malpractice. Academic institutions should make their students aware of this malpractice and implement necessary steps to prohibit students from plagiarizing. Detecting plagiarism in students' writings has always been an exhausting task for academic instructors and teachers. It is getting immensely challenging with the spread of internet usages and the easy availability of electronic documents' abundances. Typically, the most common trap students fall into is finding an article online that concerns a similar topic they have been working on, editing some parts, and submitting it. Some students have also relied on other individuals' essays from their peer groups to write their assignments. In either case, these are examples of plagiarism [1]. Fig. 1 demonstrates an example of a plagiarized work in two students' assignments. Student X copied his texts from two different sources indicated with blue and green color, whereas student Y also plagiarized from the same sources but added some words. The additional words are shown in red.

Plagiarism has an adverse effect in academia. It directly affects the learning objectives and creative thinking of students. Teachers face difficulties in grading students' assignments with a higher degree of judgement and waste their valuable time identifying plagiarism. Hence, universities and other academic institutions have accommodated several tools to identify plagiarized works and researchers are always searching for inventing robust and practical tools. Several approaches can be found in literature intended to detect plagiarism. Though the text-based yields the best result till date, those are well-suited to detect plagiarism of copy-paste type only or moderately altered text in an input document [2]. Researchers have proposed numerous approaches to improve text-similarity assessment methods by adopting methods like paraphrasing, sentence matching, or keyword matching [3]. Most of those are not freely available and freely available plagiarism checkers such as Small Seo [4], PlagScan [5], and Plagiarisma [6] impose some text limits. Moreover, none of the existing tools

978-1-6654-4923-6/21/$31.00 ©2021 IEEE

aims to serve busy academics by detecting plagiarisms within students' peer groups.

To solve the aforementioned issues, we have developed AcPgChecker that focuses on providing an effective tool for academic faculties and instructors to detect plagiarism in text-based electronic assignments. The tool throws light on the immense research potential in this field for developing efficient intelligent detection systems to curb unethical acts in students' assignments or scientific reports. Furthermore, the software can be used as a base tool for detecting plagiarism in different languages. AcPgChecker is a windows based desktop tool which provides some essentials and unique features for academic faculties and instructors -

1) The software does not require users to upload each document separately. The user only needs to browse the directory that contains all the documents.
2) The current version measures similarity among pdf documents.
3) The output is presented in CSV format containing the student identification numbers and percentage of similarity among the documents.
4) Users can customize the number of word sequences of similarity.
5) The threshold of plagiarism is also customizable depending on user needs.

We organize our rest of the work as follows. We discuss closely related existing works in Section II. After that, in Section III, we present our technique of detecting plagiarism. Then, we provide an overview of our tool in Section IV. Finally, we conclude our paper in Section V.

II. Related Work

Plagiarism is mostly identified using different similarity measurements. Thus, it is required to discuss those similarity measurement techniques. In this section, we also discuss existing plagiarism tools with their limitations.

A. Similarity Measurement Techniques

Although there are many techniques to calculate the similarity among the elements in the dataset, all of these techniques are not well fit for all types of dataset. For example, Jaccard similarity [7] works well when duplication does not matter and is good for identifying the mirror site and whereas cosine similarity [7] is good for text similarity analysis and natural language processing. Considering the advantages and disadvantages of the well established techniques, some of the techniques are described here briefly.

Euclidean distance: The length of straight line between two points in an Euclidean space is the Euclidean distance [7] [8]. It is often referred to as the Pythagorean distance because it can be determined from the Cartesian coordinates of the points using the Pythagorean theorem. If the points (x_1, y_1) and (x_2, y_2) are in 2-dimensional space, the Euclidean distance becomes

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

. The Euclidean distance is useful when measuring the distance between two points considering no obstacles between them.

Manhattan distance: The Euclidean distance fails when straight line could not be established between two points [8]. For example, measuring the distance between two streets in the real life is not straight line always. In such case, a new distance measuring technique named Manhattan distance is used [7]. The Manhattan distance calculates the distance between two points as the sum of their Cartesian coordinates' absolute differences. In other words, it is the total sum of the difference between the x-coordinates and y-coordinates. In a two dimensional coordination system, if two points are A(x_1, y_1) and B (x_2, y_2), the Manhattan distance could be calculated using the equation:

$$|x_1 - x_2| + |y_1 - y_2|$$

Minkowski Distance: It is a similarity measurement technique between two points in the vector space and is the generalized representation of the Euclidean and Manhattan distance [8]. Let's consider two data points $P_1(x_1, x_2, ..., x_n)$ and $P_2(y_1, y_2, ..., y_n)$ in a n-dimensional space, then the Minkowski distance between P_1 and P_2 is given as:

$$\sqrt[p]{(x1 - y1)^p \ + \ (x2 - y2)^p \ + \ ... \ + \ (xN - yN)^p}$$

Both Euclidean distance and Manhattan distance are same as Minkowshi Distance, because these two distance metrics are Minkowshi Distance based on the value of P. The above equation becomes Euclidean and Manhattan distance when P becomes 2 and 1 respectively.

Jaccard Similarity: This similarity measuring technique is used for finding the similarity and diversity between two sample sets [7]. For this, it measures the similarity by taking the ratio of the size of intersection divided by the size of the union of the sample sets. If A and B are two sample sets then the Similarity measure using Jaccard will be

$$JaccardSimilarity(A, B) = \frac{(A \cap B)}{(A \cup B)}$$

Although, the number of common elements tells the similarity to some extent. However, it cannot determine the degree of similarity compared to the size of the set. That is why Jaccard similarity divides the size of the intersection by the size of the union. The Jaccard similarity useful in recommendation system, measuring the mirror site, etc.

Cosine similarity: The cosine similarity approach measure the cosine angle between two vectors and determine their directions [7] [9]. It calculates the similarity between two vectors by using the ratio between dot products of two vectors and the product of these two vectors' lengths. For example, A and B both are two vectors, then the cosine similarity is calculated using the following equation. A cosine value of 0 indicates that the two vectors are orthogonal and do not fit. The smaller angle makes the cosine value closer to 1, which increases the match between two vectors.

$$CosineSimilarity(A, B) = \frac{(A.B)}{|A||B|}$$

Where,

1) A.B = product (dot) of the vectors 'A' and 'B'.
2) $|A| \ |B|$ = length of the two vectors 'A' and 'B'.

The cosine similarity works well even if the data are distributed in the large dimension because it does not calculate based on other distance measurement, for example, euclidean, rather it calculates the direction of angles to measure similarity.

B. Existing Plagiarism Tools

Turnitin [10] is a cloud based plagiarism detection tool that compares a submitted manuscript with its existing database of student work, websites, books, articles, etc. It uses both intra and outer corporal for comparing texts. It also allows both students and teachers to check for plagiarism. It usually provides text matches as well as a similarity score. Moreover, it also helps teachers evaluate student works and performance over a time through statistics and graphs. It also helps students by allowing them to learn from each performing peer reviews between their works. One of the main advantages of using this tool is that students can identify plagiarism in their manuscripts and solve these before final submission. However, this tool fails when students replace a word with similar meaning without even changing the sentence structure[10].

Eve2 is another tool that allows academicians to determine whether students have plagiarized contents from other sources or not [10]. It accepts manuscripts as plain text, or Microsoft Word and returns links to web pages from where the content have been copied. Although checking all available content in the internet is infeasible to some extent, however, Eve2 performs a set of convoluted search algorithms to find content from any Internet site. Statistics show this tools could identify 80-90% plagiarized content. Comparing with the Turnitin [10], only academicians could use this tool and it only uses outer corporal to find matches of the document. Another important advantages of using Eve2 is getting comparison results instantly.

There are other plagiarism detecting tools such as CopyCathGold and WordCheck [10]. Both of them allow only teachers or academicians to identify plagiarism in documents. Furthermore, both tools use only intra-corporal database to find match between the documents unlike Eve2 [10].

There are also some tools for detecting plagiarism in source codes. Two of them are Moss and Jplag [10] [11]. These tools identify similar source codes by comparing all the stored codes internally. The results of Moss is more detailed and easy to investigate plagiarised code. Although Moss supports more than 26 languages compared to Jplag which only supports 6 languages, these tools have lots of similar characteristics, for example, both are free to use and designed for academicians only. One of the flaws of using Moss is that it only identifies code similarity and does not tell why a particular code segment is similar. Despite having disadvantages, it is very helpful for the academicians to identify the code clone. On the other hand, Jplag does not compare from Internet rather it is only contingent on the internal source code of the students.

According to Liu et al. [12], the existing plagiarism detecting tools although performs well but fails in statement reordering. They proposed a program dependence graph based plagiarism detection techniques named GPlag which can identify plagiarism by mining program dependence graphs. It used graphic representation of the data and control dependencies within a procedure to identify plagiarism in the source code. Experimental results show that GPlag performs well in case of existing techniques' failure in statement reordering.

In summary, despite the existence of available plagiarism detecting tools, the need for academicians are still unfulfilled. Therefore, academicians would be grateful if a new effective and efficient tool is available concerning their demands.

III. Techniques of Plagiarism Detection

In order to detect plagiarism among a set of documents (assignments, reports and etc.), we employed a well-known Information Retrieval (IR) technique - Vector Space Model (VSM) [13]. The following subsections discuss the required concept for fitting a set of documents into VSM. For explaining each concept clearly, we use the following two documents as an example throughout this section.

- **Document #1**: A survey of user opinion of computer system response time.
- **Document #2:** Relation of user response time to error measurement.

A. Representing documents as a vector

A document can be represented as a vector with/without preprocessing (removing stopwords, punctuations, special characters). It is worth noting that preprocessing should be done before applying any IR technique [14]. If Document #1 and Document #2 are represented as vectors after preprocessing, they will look like the following.

- **Vector #1:** (survey, user, opinion, computer, system, response, time)
- **Vector #2:** (relation, user, response, time, error, measurement)

B. Document-Term Matrix

A document-term matrix is an alternative way of representing a set of documents in a matrix form; each row represents unique documents and each column represents the unique terms/words across the all documents. The cell values represent the frequencies of each term in the corresponding document. For Document #1 and #2, the document-term matrix is displayed in Table I. In this example (Table I), a single word is considered as a term. However, multiple adjacent words can be treated as a term. The main idea of this way is that if adjacent n words are found same in several documents then it will be considered as plagiarism. In this study, we use multiple adjacent words as a single term. For detecting plagiarism among several documents, it is required because people used to copy & paste adjacent words [15]. Although document-term matrix is considered as a nice way of representing documents, it does not provide any weight to

the terms/words. The contribution of each term/word may not be equal in terms of document relevance or similarity [16]. Hence, it is suggested to calculate the weight of the terms and then term-document matrix should be generated [17].

C. Term Frequency (TF)

Term Frequency (TF) refers to the total occurrences of a particular term in a particular document. It is denoted by tf_{tD} where t is the term and D is the document. If D = Document #1 and t = user, the value of $tf_{tD}(user)$ is 1.

D. Document Frequency (DF)

Document Frequency refers to the total number of documents that hold a particular term (t). It is denoted by df_t. For the term "response", the value of df_t is 2 because both Document #1 and #2 hold the term.

E. Inverse Document Frequency (IDF)

Inverse Document Frequency (IDF) discriminates each document based on the uniqueness of a term. It is computed using the Equation 1 where N is the number of documents.

$$idf_t = \log_2(N/df_t) \tag{1}$$

F. Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) is calculated by combining the TF and IDF of each term. The purpose of this TF-IDF is to assign a weight to the terms based on their importance. TF-IDF of a particular term is calculated as follows.

$$tf - idf = tf_{tD} \times idf_t \tag{2}$$

G. Document-Term Matrix using TF-IDF

Since TF-IDF provides a weight to the terms based on their importance, it is suggested to generate a document-term matrix using TF-IDF. For Document #1 and #2, the revised document-term matrix using TF-IDF is displayed in Table II. For N number of documents, the size of document-term matrix M_{tf-idf} will be $N \times t$. Here, t represents the number of unique terms across the all documents. Besides, the rows of M_{tf-idf} are the vector representation of the documents.

H. Similarity Measurement

After generating document-term matrix using TF-IDF, the next step is to measure similarity between documents. Since the rows of M_{tf-idf} represent documents in vector form, we can identify plagiarism among document by measuring similarities among the rows of M_{tf-idf}. Let, D_i^v and $D_{i'}^v$ are the vector representation of two documents such as D_i and $D_{i'}$, respectively. Then, the similarity between D_i and $D_{i'}$ can be calculated using Cosine Similarity [18]. In this study, we use Cosine Similarity because it performs better than other similarity measurements (discussed in the Section II). The formula of Cosine Similarity is given below. Finally, if the similarity between two documents is found to be greater than a defined threshold (e.g., 30%) then it will be considered as plagiarism.

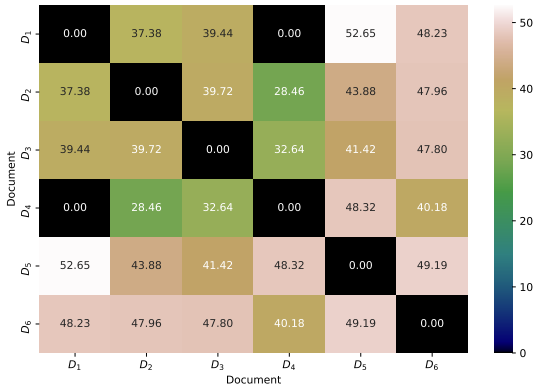

Fig. 2: Sample Plagiarism Report Between six documents

$$cos\_sim(D_i^v, D_{i'}^v) = \frac{D_i^v \cdot D_{i'}^v}{\|D_i^v\| \|D_{i'}^v\|} \tag{3}$$

As above Formula 3 calculates similarity/plagiarism between two documents only, we propose Algorithm 1 for identifying plagiarism among a set of documents (e.g., $D_1 \dots D_N$). Our algorithm takes list of documents ($D_1 \dots D_N$) as input and it outputs a plagiarism report ($Report[N, N]$). After receiving the $DocumentList[N]$, it generates TF-IDF matrix $M_{tf-idf} \in N \times t$ (Line #2). Here, N is the number of documents or rows and t is the number or columns of unique terms. Then, another matrix $Report[N, N]$ is initialized to store the similarities among the N number of documents (Line #3). To find similarity between documents, we use two loops that iterate row-wise over the M_{tf-idf} (Line #4 & #5). To avoid calculating similarity between same documents, a condition in Lines #6 to #9 is used. In Lines #10 to #12, similarity between different documents is measured. Then, the calculated $Similarity$ (Line #12) is compared with a predefined $Threshold$ in Line #13. If $Similarity$ is greater than $Threshold$, $Similarity$ is stored in the $Report$ matrix (Line #14). Otherwise, no similarity is stored in the $Report$ (Line #16). Lastly, the final $Report$ is returned in the Line #20. A sample plagiarism report is displayed in the Fig. 2. Each cell of this figure represents the similarity/plagiarism between two documents. For instance, document D_3 is 41.42% similar with D_5.

IV. Plagiarism Detection Tool

To validate our idea, we develop an open source tool which is coined as AcPgChecker. The current version of our tool is developed for Windows platform. For developing the User Interface (UI), we use Windows Presentation Foundation (WPF) [19] UI framework. In back-end, we use C# [20] programming language. The AcPgChecker V1.0[1] is released

[1]https://github.com/dipongkor/PlagiarismChecker/releases/tag/v1.0

TABLE I: Document-Term Matrix obtained from Document #1 and #2

a	computer	error	measurement	of	opinion	relation	response	survey	system	time	to	user
1	1	0	0	2	1	0	1	1	1	1	0	1
0	0	1	1	1	0	1	1	0	0	1	1	1

TABLE II: Document-Term Matrix using TF-IDF

a	computer	error	measurement	of	opinion	relation	response	survey	system	time	to	user
0.3421	0.3421	0.000	0.0000	0.4868	0.3421	0.0000	0.2434	0.3421	0.3421	0.2434	0.0000	0.2434
0.0000	0.0000	0.4074	0.4074	0.2898	0.0000	0.4074	0.2898	0.0000	0.0000	0.2898	0.4074	0.2898

Algorithm 1 Detection of plagiarism from a set of documents

Require: $D_1 \dots D_N$ (List of Documents)
Ensure: $Report[N, N]$ (Plagiarism Report)

function GetPlagiarismReport($DocumentList[N]$)
  $M_{tf-idf} \leftarrow$ tf_idf_matrix($DocumentList[N]$)
  $Report[N, N] \leftarrow \emptyset$
  for $row \leftarrow 1$ to M_{tf-idf} **do**
    for $row' \leftarrow 1$ to M_{tf-idf} **do**
      if $row == row'$ **then**
        $Report[row, row'] \leftarrow 0$
        $Continue$
      end if
      $V_{row} \leftarrow M_{tf-idf}[row]$
      $V_{row'} \leftarrow M_{tf-idf}[row']$
      $Similarity \leftarrow cos\_sim(V_{row}, V_{row'})$
      if $Similarity > Threshold$ **then**
        $Report[row, row'] \leftarrow Similarity$
      else
        $Report[row, row'] \leftarrow 0$
      end if
    end for
  end for
  return $Report[N, N]$
end function

and it is widely being used in the Jashore University of Science and Technology. In the next subsections, we discuss guidelines for end-users and contributors so that our tool becomes user friendly.

A. Documentation for End-Users

In this section, we discuss the guidelines for end-users such as students and faculty members so that they can use our tool easily. At first, users need to download our tool from here[2]. Then, it is required to install and run the application. If the application runs successfully, the user-interface as displayed in Fig. 3 will be displayed. Finally, users need to provide the following inputs (as shown in Fig. 3) for identifying plagiarism among documents.

1) **Folder location:** A directory location that contains all the documents of whose plagiarism will be measured. The current version of our tool only supports pdf documents.

[2]https://bit.ly/38zmIt5

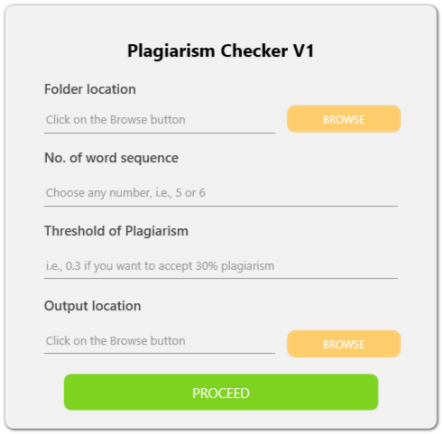

Fig. 3: User-Interface of the Plagiarism Detection Tool

2) **No. of word sequence:** Any positive number n. If adjacent n words are found same in several documents, it will be considered as plagiarism.
3) **Threshold of Plagiarism:** A threshold value of plagiarism. If similarity between two documents is less than this threshold, it will not be considered as plagiarism.
4) **Output location:** A directory location for storing the plagiarism report. Currently, our tool generates a csv file that contains the similarity/plagiarism (%) among documents.

After providing above inputs, users need to click on the PROCEED button. Lastly, users can view the plagiarism report by browsing the output location.

B. Documentation for Contributors

Since our plagiarism tool is fully open-source, we encourage contribution from the communities. We hope that community contributions can enrich our tool day-by-day. The guidelines for contributors are given below.

1) Download Visual Studio 2019 Community Edition[3].
2) Install .NET 4.5 while installing Visual Studio 2019
3) Clone the repository from GitHub[4]
4) Build the Solution. It will automatically add dependencies from the NuGet[5]

To make our project more maintainable, we break down our whole solution into six projects/class libraries. Moreover, this

[3]https://visualstudio.microsoft.com/downloads
[4]https://github.com/dipongkor/PlagiarismChecker.git
[5]https://www.nuget.org

break-down will ensure single responsibility [21] in the code level. The brief description of our projects are given below. Here, each project has its own unit-tests project so that new changes in the code does not break existing functionalities.

- **Plagiarism.Vectoriser:** It is a class library for creating n-gram and document-term matrix. The unit tests of this library are written in **Plagiarism.Vectoriser.Tests**
- **PlagiarismChecker.Pdf.Parser:** This class library parses text from PDF documents. The unit tests of this library are written in **PlagiarismChecker.Pdf.Parser.Tests**
- **PlagiarismChecker.Ui:** It is a Windows Presentation Foundation (WPF) [19] solution which implements the UI and functionalities
- **Plagiarism.Setup:** It is the setup project which generates installer file. This project has a dependency with an extension named Microsoft Visual Studio Installer Projects. It is required to install this extension before running the project.

C. Limitations and Future Improvement Scopes

Although current version of plagiarism detection tool is quite usable, there are still scopes for improvements. We aim to add the following features to our tool in future.

1) Current version considers only textual information for measuring similarities among documents. It is also found that students also copy figures from each other. Thus, it can be an useful feature.
2) Current version provides total similarities among documents only. It does not indicate exact location or similar line numbers. We aim to add this feature in future.
3) Current version only supports Windows platform. Thus, it is required to develop a cross-platform application to support Linux and Mac.

V. Conclusion

Detecting plagiarism in academic writings is a very important task due to identify the originality and the novelty of student's work. Although there exist many tools, for example, Turnitin, Eve2, etc., none of the tools detect plagiarism within students' peer groups. In this study, we proposed a new tool named AcPgChecker which providing an effective way to the academicians by detecting plagiarism in electronic documents (e.g., assignments, thesis, and other papers submitted by students). AcPgChecker uses cosine similarity to identify plagiarism in the student's work by comparing with provided all documents. This tool is much more effective than other tools considering its provided customization abilities, for example, academicians can customize the length of a sequence in the longest common word's sequence in multiple documents. Furthermore, Academicians can even change the threshold value of detecting plagiarism considering their needs. Considering the benefits of AcPgChecker, it is needless to say that it is better and convenient for the academicians.

Acknowledgement

We wish to appreciate the Department of Computer Science and Engineering (CSE), Jashore University of Science and Technology (JUST). We conduct all experiments in the Data Science and Machine Learning (DSML) lab of CSE, JUST.

References

[1] M. S. Anderson and N. H. Steneck, "The problem of plagiarism," in *Urologic Oncology: Seminars and Original Investigations*, vol. 29, no. 1. Elsevier, 2011, pp. 90–94.

[2] D. Weber-Wulff, *False feathers: A perspective on academic plagiarism*. Springer Science & Business, 2014.

[3] D. Gupta *et al.*, "Study on extrinsic text plagiarism detection techniques and tools." *Journal of Engineering Science & Technology Review*, vol. 9, no. 5, 2016.

[4] smallseotools, "Plagiarism checker - 100% free online plagiarism detector," smallseotools.com, uRL:https://bit.ly/3xLd5lC. [Online]. Available: https://bit.ly/3xLd5lC

[5] PlagScan, "Online plagiarism checking — plagscan," www.plagscan.com, uRL:https://www.plagscan.com/en. [Online]. Available: https://www.plagscan.com/en

[6] plagiarisma, "Free online plagiarism checker for teachers and students," plagiarisma.net, uRL:http://plagiarisma.net. [Online]. Available: http://plagiarisma.net

[7] W. H. Gomaa, A. A. Fahmy *et al.*, "A survey of text similarity approaches," *International Journal of Computer Applications*, vol. 68, no. 13, pp. 13–18, 2013.

[8] J. Arora, K. Khatter, and M. Tushir, "Fuzzy c-means clustering strategies: A review of distance measures," *Software Engineering*, pp. 153–162, 2019.

[9] R. Islam, A. Satter, A. K. Dipongkor, M. S. Siddik, and K. Sakib, "A novel approach for converting n-dimensional dataset into two dimensions to improve accuracy in software defect prediction," vol. 15, no. 6, pp. 147–162, 2020.

[10] R. Lukashenko, V. Graudina, and J. Grundspenkis, "Computer-based plagiarism detection methods and tools: an overview," in *Proceedings of the 2007 international conference on Computer systems and technologies*, 2007, pp. 1–6.

[11] J. Hage, P. Rademaker, and N. van Vugt, "A comparison of plagiarism detection tools," *Utrecht University. Utrecht, The Netherlands*, vol. 28, no. 1, 2010.

[12] C. Liu, C. Chen, J. Han, and P. S. Yu, "Gplag: detection of software plagiarism by program dependence graph analysis," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 872–881.

[13] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[14] J. Camacho-Collados and M. T. Pilehvar, "On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis," pp. 40–46, 2019.

[15] H. A. Maurer, F. Kappe, and B. Zaka, "Plagiarism-a survey." *J. UCS*, vol. 12, no. 8, pp. 1050–1084, 2006.

[16] A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, 2003.

[17] G. G. Chowdhury, *Introduction to modern information retrieval*. Facet publishing, 2010.

[18] S. Tata and J. M. Patel, "Estimating the selectivity of tf-idf based cosine similarity predicates," *ACM Sigmod Record*, vol. 36, no. 2, pp. 7–12, 2007.

[19] Microsoft, "Get started with wpf," docs.microsoft.com, uRL:https://bit.ly/36EzjcY (version: 2016-10-13). [Online]. Available: https://bit.ly/36EzjcY

[20] ——, "C# documentation," docs.microsoft.com, uRL:https://bit.ly/3hHM9xw (version: 2016-10-13). [Online]. Available: https://bit.ly/3hHM9xw

[21] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.