mathematics

MDPI

Article

DocCompare: An Approach to Prevent the Problem of Character Injection in Document Similarity Algorithm

Anupama Namburu [1], Akhil Surendran [1], S Vijay Balaji [1], Senthilkumar Mohan [2,*] and Celestine Iwendi [3]

[1] School of Computer Science and Engineering, VIT-AP University, Andhra Pradesh 522237, India
[2] School of Information Technology and Engineering, Vellore Institute of Technology, Vellore 632014, India
[3] School of Creative Technologies, University of Bolton, A676 Deane Rd., Bolton BL3 5AB, UK
* Correspondence: senthilkumar.mohan@vit.ac.in

Abstract: There is a constant rise in the amount of data being copied or plagiarized because of the abundance of content and information freely available across the internet. Even though the systems try to check documents for the plagiarism, there have been trials to overcome these system checks. In this paper, the concept of character injection is used to trick plagiarism checker is presented. It is also showcased that how does the similarity check algorithms based on k-grams fail to detect the character injection. In order to eradicate the problem or error in similarity rates caused due to the problem of character injection, image processing based approach of multiple histogram projections are used. An application is developed to detect the character injection in the document and produce the accurate similarity rate. The results are shown with some test documents and the proposed method eliminates any kind of character injected in the document that tricks plagiarism. The proposed method has addressed the problem of character injection with image processing based changes in the existing methods of document-similarity check algorithms using k-grams. The proposed method can detect 100% injected character be it any alphabet of any language, The processing time for conversion, histogram projections and applying winnowing algorithm takes 1.2 sec per page on average when experimented on multiple types of document varying in size from 2 KB to 10 MB.

Keywords: plagiarism detection; character injection; image segmentation; word segmentation; histogram projection; document analysis

MSC: 68U10, 94A08

Citation: Namburu, A.; Surendran, A.; Balaji, S.V.; Mohan, S.; Iwendi, C. DocCompare: An Approach to Prevent the Problem of Character Injection in Document Similarity Algorithm. *Mathematics* **2022**, *10*, 4256. https://doi.org/10.3390/math10224256

Academic Editors: Natalia Kryvinska and Michal Greguš

Received: 18 October 2022
Accepted: 9 November 2022
Published: 14 November 2022

1. Introduction

Digital content has seen a massive growth in the recent years. Especially, at current fastest hi-tech virtual world, professional institutions and educational institutions are moving their resources and services online. In early 2020 with the outbreak of of COVID-19, teaching-learning suddenly went online, making life digital by default. In the study [1], how the pandemic has caused a surge in the adoption of digital technologies in areas and it was slowly progressing and the increase in the amount of data flowing through the internet is explained. Abundant of data and information is freely available on the internet, plagiarism has taken a toll on the quality of submissions being made. Checking for multiple documents manually takes a good amount of time as well as labour and for that same reason, there are multiple software systems that are used to automate the process and check for similarity.

Plagiarism is a critical issue for researchers and the reputation of the educational institutes rely on the quality of publications [2]. Especially in the present scenario, the outbreak of COVID-19 pandemic has highlighted the true dimensions and importance of the internet in conducting educational processes [3]. For the same reason, the requirement of strong checks are necessary in order to maintain the quality and credibility of submissions being made.

Plagiarism is generally classified as textual and source code plagiarism [4]. The former is more commonly observed in scientific and research field taking their text or document with out citing or quoting them. These plagiarism include copy-paste, paraphrasing like original text replacing with new words or summarizing the content from two three documents with out referencing them, copying the idea, presenting it in a better way, using old or existing work to present the current research, re-tweeting like which shows no difference between the existing and present work in terms of structure, grammar and words [5]. Source code plagiarism is mainly observed in educational institutes, where the programming code is being plagiarised. This plagiarism include manipulation of code, reordering the blocks or functions used, changing the spaces, comments in the Code and language switching of the program [6,7].

Many researchers and software development industry started working on the detection of plagiarism. The detection of plagiarism in a document is verifies if the content is modified to other language or modifying the content itself. If the Content is translated to other languages the plagiarism detection is performed using cross language similarity method [8–13]. If content is being modified in the document, plagiarism is performed with intrinsic and extrinsic approach. Intrinsic plagiarism is performed to identify the instances of plagiarism with in the query document itself. Extrinsic plagiarism uses reference documents for checking the plagiarism [14]. To perform extrinsic plagiarism, it uses the principle of finger prints of text. In the finger print approach, the input document and reference documents are separated to small chunks named as fingerprints (or shingles) of length k. These Fingerprints are used to generate the hash values (or hash count) with the help of a hash function. The hash values are generated for each documents vectors and are unique for every document. Further, the hash values generated for the input document and the reference document are matched. The similarity measures between the documents shows the level of plagiarism in the document [15–17]. Many plagiarism tools were developed to detect the plagiarism based on fingerprints.Docol©c [18] Ithenticate [19], Turnitin, Urkund,Copycatch,WCopyfind,Plagiarism Detect,Exactus Like,DupliChecker,Plagiarisma, Plagiarism Checker, Viper and Plagiarism Scanner [7,20,21] are different software tools to check textual plagiarism based on finger prints. These tools are good at detecting the plagiarised content efficiently.

However, in recent years, a new way of cheating the plagiarism tools came into lime light namely, character Injection technique. In this scenario, a character is added at the end of words, and its font is minimized and converted to white color. This will hide the characters at the end of each word [22]. With this hidden characters, if plagiarism tools are used to find the similarity then the result in inaccurate and shows less plagiarism which actual is not true. In order to address this problem a novel approach using image segmentation based on histogram projections is used to detect the plagiarism based on character Injection.

The problem of character injection, how document similarity works and histogram projections are explained in the background in Section 2. The solution with image segmentation with histogram projection is presented in the proposed methodology in Section 3. The results and experimental discussion is presented in Section 4. The conclusion and the future scope are presented in Sections 5 and 6.

2. Background

2.1. Problem of Character Injection

Definition: A technique, used to trick plagiarism detection systems that uses document similarity algorithms using k-grams wherein characters are injected into the documents at various intervals and masked in a way that it is invisible to the viewer and on extracting text, these characters tend to create errors in the similarity rates.

One of the gram based algorithms used to identify plagiarism is Winnowing [15]. Winnowing algorithm is a method of word similarity search in a document by comparing the fingerprints on the document [15]. The algorithm uses the method of k-gram parsing,

which is then converted into corresponding hashes resulting into unique fingerprints. K-grams are converting a particular document or string to a sub-string sequence, with each sub-string having a length equal to *k*. [15]. The problem that is highlighted in this paper is the character injection as shown in Figure 1, when characters are being injected into the documents at regular intervals. For instances, when two document with same text-say document A and document B we considered. We replace all the white spaces in document B with a new character, this changes the entire document.

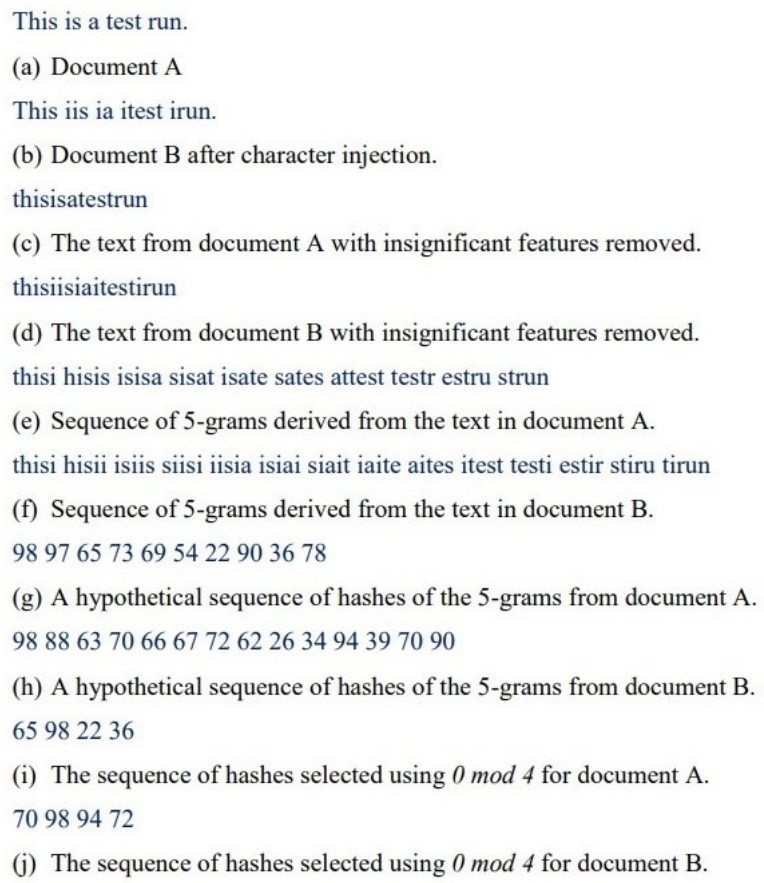

Figure 1. Winnowing with character injection.

Consider a situation represented in Figure 1, where two documents A and B are exact same when the content is considered, but it is observed ' ' have been replaced by a white space and a character, *i* in this case. Now, when the k-grams parsing is performed after the prepossessing stage, the k-grams generated as shown in Figure 1, are different from one another for documents A and B. When these K-grams are converted to their corresponding hashes and when the hashes are identified for fingerprints, the resultant similarity rate drops down indicating both these documents to be different, while that is not the case. In the following sections, a novel method is proposed to overcome this problem.

2.2. Document Similarity Algorithms

The document similarity algorithms involve some prior requirements in order to be applied to the documents. These primary requirements that are supposed to be completed by the algorithm are as follows: [15]

- Insensitivity due to white spaces, the process of search should not be induced by white spaces, capitalization, punctuation, etc.
- Noise elimination, short matches should be avoided, for eg. finding words like the, is, etc., is not desired.

- Position independence, the document analysis for checking similarity should not be dependent on the position of a word, as such that moving around of words in a sentence or paragraphs across the document would not affect the test.

The document-similarity check based on the Winnowing algorithms are put across in the following steps: [23]

- Considering target text and original text as two strings.
- Processing of these texts, removing of white spaces, punctuation, etc. as specified by the algorithm.
- Parsing the document into k-grams. k-gram corresponds to a sub string that has a length equal to the value of k, and the parameter k can be specified by the user.
- Generating hashes for each grams.
- Identifying some hashes from the available set as the document fingerprint.

This process mentioned above is not an attempt to remove the injected characters that have a possibility of being introduced into the document in order to trick the system. Such characters when introduced will have bias on the detection systems in such a way that, the hashes being generated will be unique, making the fingerprints unique and this will result in the deterioration of the similarity rates.

2.3. Histogram Projection

Histogram projection is a technique mainly used to extract the characters from images [24]. Histogram is a plot between the intensity of the pixel and the count of the intensity. The plot is drawn for each pixel in the image for its count of times appeared in the image. Histogram projections are the extended version of histograms, where the plots are drawn for row/columns of the image and the count of ON pixels(representing character) in that row/columns. Mostly histogram projections are performed on binary images. The Figure 2 shows the horizontal $HP_H[i]$ and vertical count $VP_H[j]$ of pixels computed using the Equations (1) and (2).

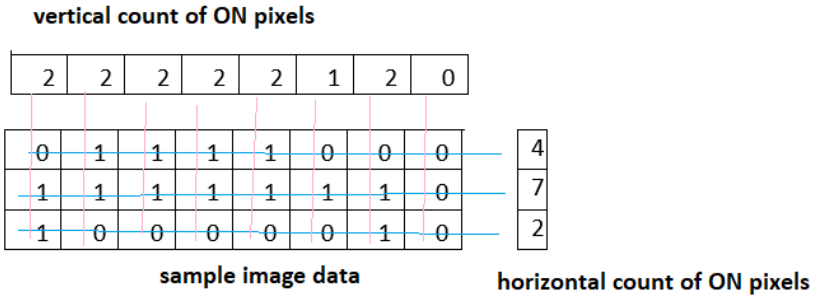

Figure 2. Vertical and horizontal count of pixels.

$$HP_H[i] = \sum_{j=0}^{M-1} I(i,j) \quad \forall I(i,j) = 1 \tag{1}$$

$$VP_H[j] = \sum_{i=0}^{N-1} I(i,j) \quad \forall I(i,j) = 1 \tag{2}$$

$$plot(i, HP_H[i]) \quad 0 <= i <= M-1 \tag{3}$$

$$plot(j, VP_H[j]) \quad 0 <= j <= N-1 \tag{4}$$

Here, M represents the rows and N represents the columns of the image. $I(i,j)$ represents the intensity value of the image and it is is 1 it indicates ON pixel. The histogram horizontal projection is the plot between rows of the image and count of ON pixels in each row as shown in Equation (3).

In vertical projections for each column, the count of ON pixels are plotted as shown in Equation (4). The horizontal projections are useful to segment the lines while the vertical projections are useful in extracting the words and characters.

2.3.1. Horizontal Projections for Line Segmentation

In horizontal projection for each row, the number of ON pixels are plotted. Here, the horizontal projections are applied to the document that are converted to an image. For the converted image, for every row, the number of ON pixels are plotted as shown in the Figure 3. Here, the projections image is rotated in order to make the user understand the projection in comparison to the text document. Irrespective of the language of the document, each peak in the histogram projection represents one line of the image document. Using this peaks each line position is identified and segmented. The lines are further processed with vertical projections to extract words and characters.

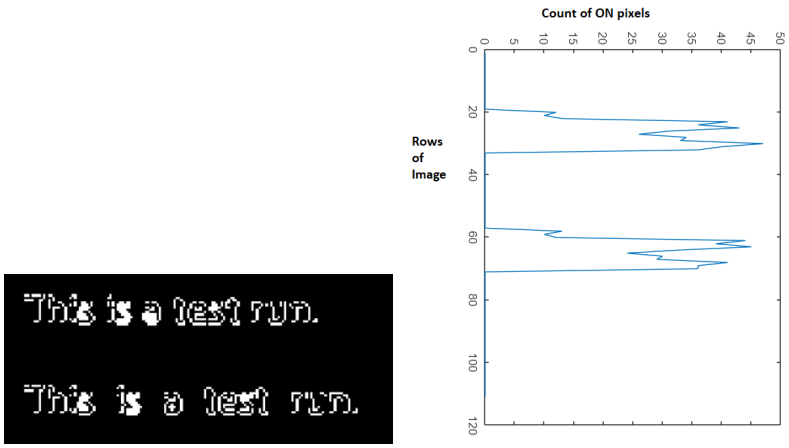

Figure 3. Horizontal histogram Projection and histogram projection of each line.

2.3.2. Vertical Projections for Word Segmentation

In vertical projections for each column, the number of ON pixels are plotted. Each peak in the vertical projection represents one word in the line. Based on the column position the words are extracted. Figure 4 represents the vertical projections of the line.

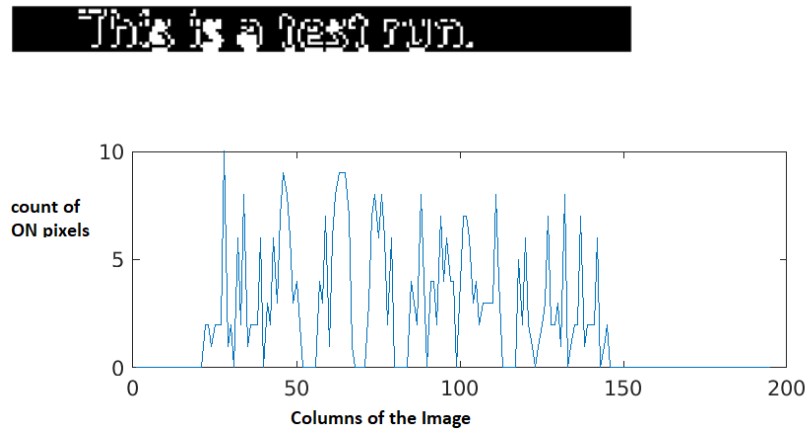

Figure 4. Vertical Histogram Projection-Line.

3. Proposed Methodology

As shown in Figure 5, the first step to this approach is uploading the document. In the next step, image segmentation is performed on the converted document(image). In order to make this work universal in case of different document types, we convert the uploaded

file irrespective of the file type such as .docx, .pdf, etc., to image type files. Once we convert the file to corresponding image files, we then perform the process of segmentation on these images. As a result of the segmentation process, we receive the extracted text from the images using histogram projections.

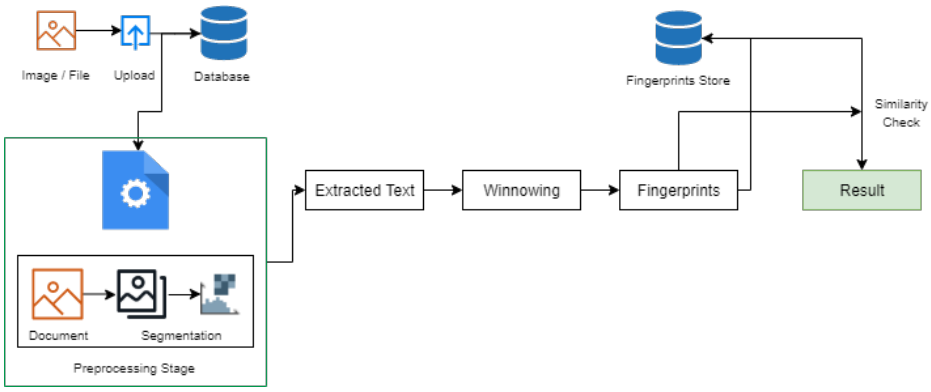

Figure 5. Steps performed in this experiment.

In order to check the similarity of the documents, generally, the Winnowing algorithm [15] is used that works on basis of k-grams approach. Finally, the obtained hashes from the finger prints are matched with the reference document and then the similarity is resulted. However, this approach fails to detect the character injection in the document and the similarity rate is altered. In order to address this problem, the image segmentation process is added to the existing winnowing algorithm highlighted in green in Figure 6.

The document for which the similarity is to be identified, is converted to image first and then lines are extracted from the image. Further, the vertical histogram projections are applied on the lines to finds the peaks of the words and then the spaces are removed and extract only the characters.

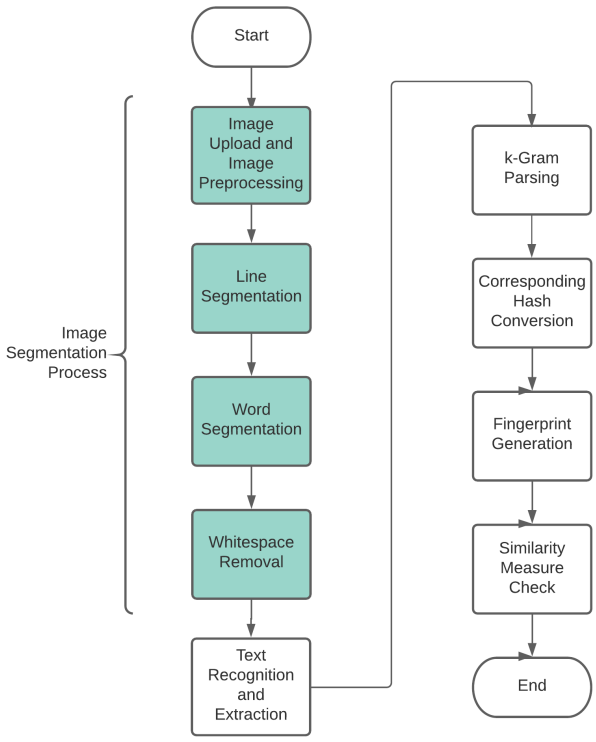

Figure 6. Proposed Similarity Check-Process Flow.

3.1. Character Segmentation Based on Histogram Projection

For the purpose of character segmentation from the image documents, an approach of character segmentation using multiple histogram projection is used as specified in study [24]. The process consists of a sequence of steps composing a complete pipeline [25], which is divided into two parts-pre-processing stage and text recognition.

Image document pre-processing plays a vital role in influencing the performance during the recognition phase. Therefore, it is divided into the following parts:

- Binarization [26]
- Noise reduction

In binarization, the image is converted to binary form having white background with black characters. Each black pixel is treated as ON pixels. The binary images are helpful in extracting the histogram projections. The Noise reduction need to be performed on the document to remove stop words. Noise reduction is performed in document by removing symbols $,,'$, $[]!, ?, :, ;$, $-$. On completion of the pre-processing stage, further the segmentation process is applied. Here, the histogram projections are obtained in order to perform line segmentation followed by word segmentation, as explained in [24].

3.2. Proposed Plagiarism Detection

The proposed method presents two features, similarity check and character injection test. Similarity check feature is used to detect plagiarism within the document A in reference to other documents. The Character injection test is used to detect plagiarism within the same document in presence of character injection.

3.2.1. Feature 1-Similarity Check

Figure 7 represents the process flow for the first feature-the document similarity check. For this experiment, two documents are compared at a time. As mentioned in the flow of the Figure 7, a text extraction from document A is performed at the same time document A is also converted to image and based on the process explained in Figure 6, the text is extracted using histogram approach. Now, on the extracted text with and with out image processing, k-gram parsing , corresponding hash conversions, fingerprint generation is performed. On comparing the fingerprints, if both of them are producing same fingerprints then character injection is not found else character injection is found. Similarly for document B the same process is performed. When both document A and document B's hashes are matched and no character injection then similarity report is generated. If document A and document B has character injection then the character injection is removed and compared for similarity. This way the character injection in documents that tricks plagiarism check can be eliminated. This process can be scaled up easily and provisioned for multiple documents for the future.

3.2.2. Feature 2-Character Injection Test

The same Figure 7 is applicable for the process flow of the second feature-the character injection test except that the input is only one document. This detects the presence of injected characters in a single document and notifies the level of injection as a part of the results. Here, in this case as seen in the process flow (Figure 7) the hashes generated directly with winnowing and the hashes generated after image processing, are compared. If both are same then there is no character injection else there is a character injection.

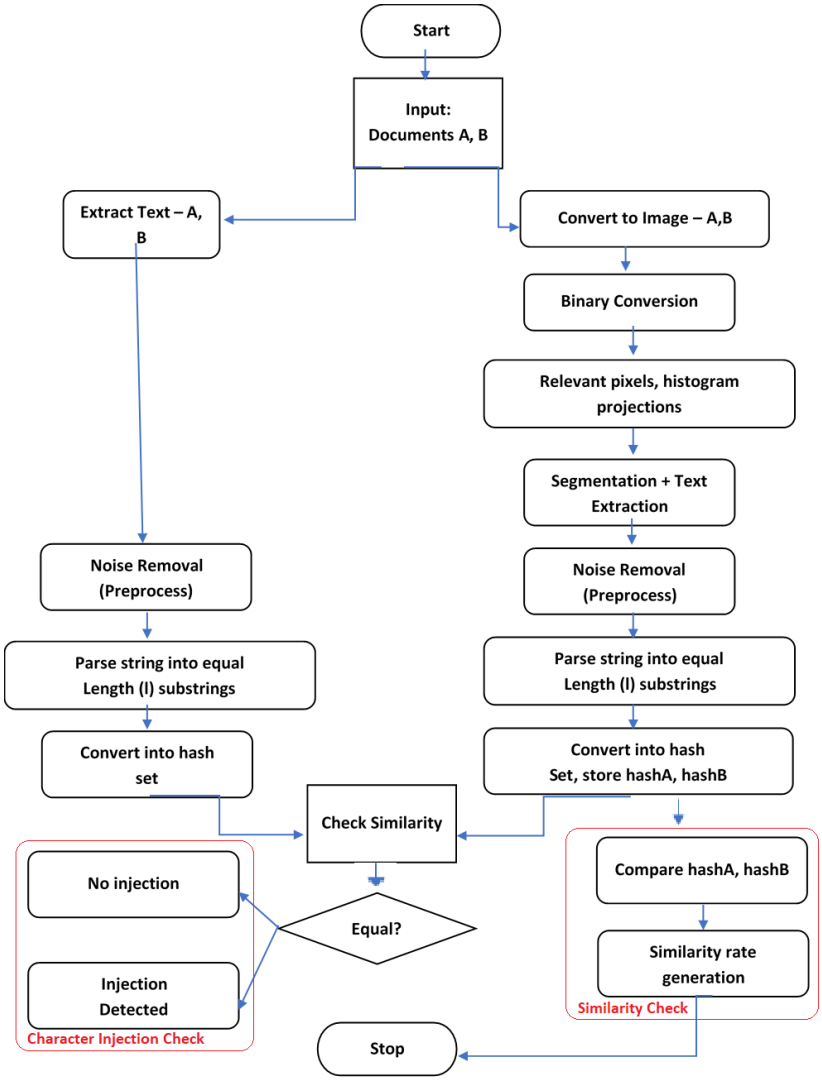

Figure 7. DocCompare-Document Similarity Check (Process Flow).

4. Results and Discussion

The experimental setup of the proposed model is shown in this section. The experiment is executed on Intel i7 processor with 16GB GPU and it is executed using Python. The documents are converted to images and then histogram projections are applied. The Machine learning algorithm winnowing [15] is used for fingerprint and hash keys generations. To explain the analysis of the proposed model a small text document is considered as shown in Figure 8.

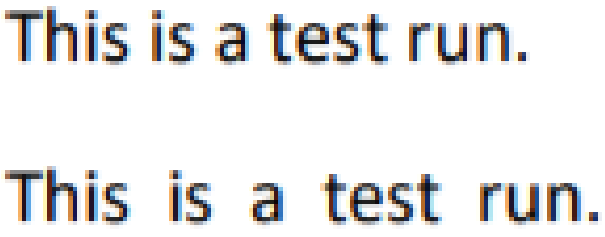

Figure 8. Test Image Document.

The first line of the document is the normal text while the second line is the hidden character injected document. A fair amount of difference is observed in the space between words, when these text lines, line1 and line2 are compared. In Figure 8, Line 2 shows that character 'i' is injected before every word and it's font is changed to white color, hence making the character 'i' invisible in the Line 2. However, the space difference can be observed between the words. Hence, the plagiarism software fails to detect the plagiarism as they do consider the hidden character in the process. Hence, to avoid this problem, in this approach the document is converted to image and then the characters are extracted that can be further used for plagiarism checking. From extracted projections in Figure 3, each line is extracted. After the pre-processing stage, pixel based histograms are plotted as mentioned in algorithm proposed in the study [24] and these histogram projection shown in Figure 3 are considered for the Line segmentation process [24], as shown in Figure 3. Once the line segments are obtained, the segmentation of words from these line segments need to be performed.

The histogram projection for line 1 is shown in Figure 9 and for line 2 in Figure 10. It can be clearly observed that the graph for line 2 shows a bigger interval difference for white spaces when compared to that of line 1, this difference denotes the masked characters along with the white spaces in line 2. As the next step, the white spaces are eliminated and the resultant graphs are shown in Figures 11 and 12 for lines 1 and 2 respectively. From the Figures 11 and 12, it can be clearly observed that both line 1 and line 2 after removing the spaces the projections are not the same, this can be seen in Figure 13 proving that there is alteration of the content with character injection. The advantage of converting into image and then extracting the character is that, the masked character is invisible in image and when we extract the text from image, only the visible characters are extracted, hence avoiding the character injection (hidden characters). Further, the text extracted from the images after elimination of injected character will further undergo the process of similarity detection.

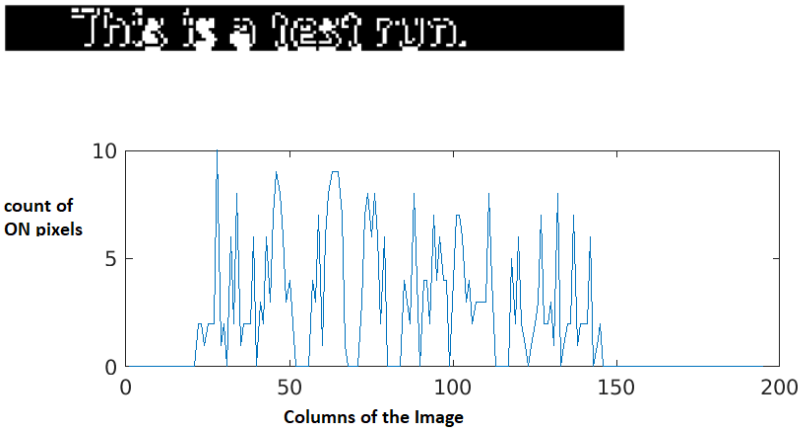

Figure 9. Histogram Projection-Line 1 (whitespaces included).

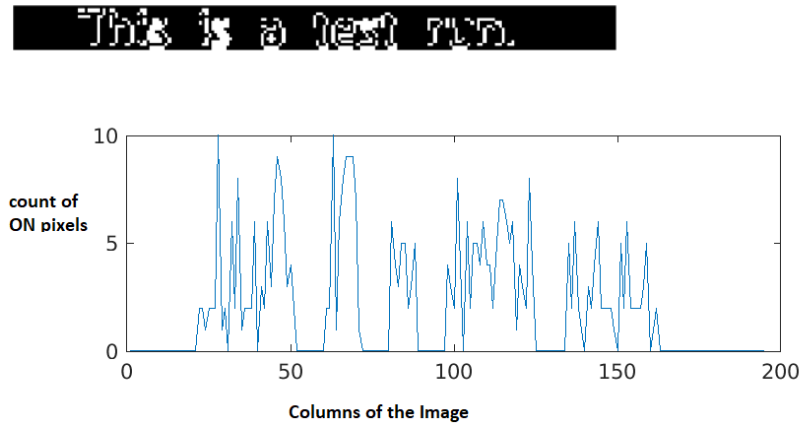

Figure 10. Histogram Projection-Line 2 (whitespaces included).

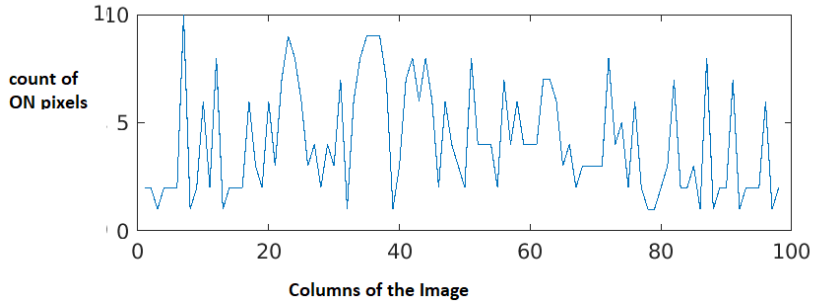

Figure 11. Histogram Projection-Line 1 (white spaces removed).

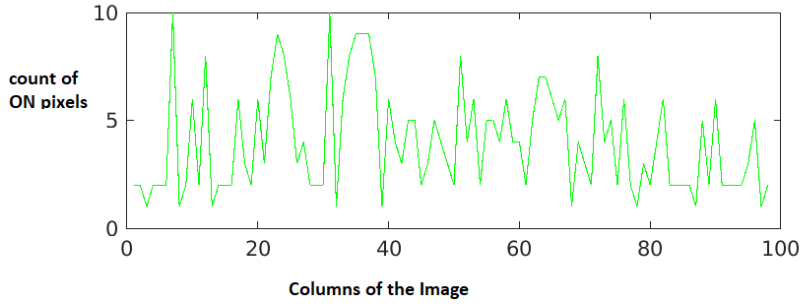

Figure 12. Histogram Projection-Line 2 (white spaces removed).

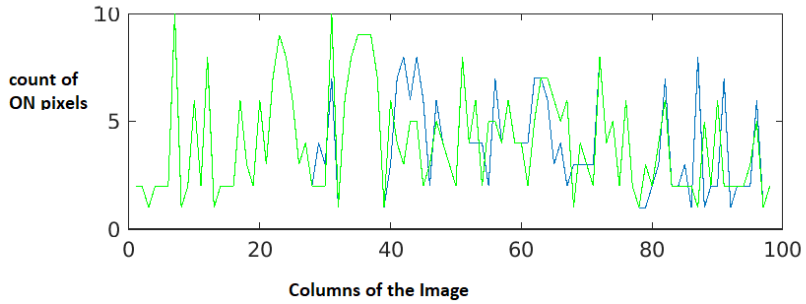

Figure 13. Overlapped results of histogram projections of line 1 and line 2.

Application: DocCompare

For the purpose of this experiment and to demonstrate the technique mentioned above, an application portal called DocCompare is created.

Figure 14 shows the Home Page section of the application which provides the option and details to the features available.

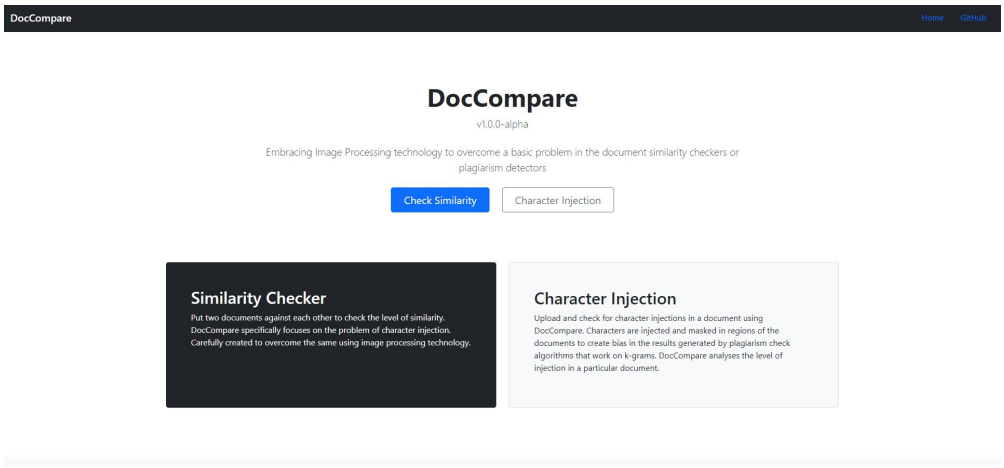

Figure 14. DocCompare-Application Home Page.

This application performs the two main features (checks)—the first one being document similarity check on multiple documents and the second one is-the character injection test on a single document. The app created for the document similarity check overcomes the problem of injected characters and thus reports a result, unbiased, caused due to the injection. The character injection (single document) test measures the level of injection in the uploaded document. Once the process of similarity check is completed, the results are available in the form of a .pdf report that can be generated using the "Generate Report" option in the Results section.

- File Details (for both uploaded documents)
  - File Name and Type
  - File Upload Date and Time
  - File Size
  - File author
- Results Section
  - Similarity Rates
  - Generated grams with their corresponding hashes

The sample outputs for comparing document 1 and document 2 are shown in the Figure 15. Document 2 contains same text except that character 'i' is injected before every word. The App output clearly highlights the injected characters as well as the similarity score is 100% as both the documents have same text.

In order to measure the similarity index between the documents the following coefficients are used.

Jaccard's Coefficient:

Jaccard's coefficient [27] compares members for two sets to determine which are shared and which are distinct. Niwattanukal et al. explains in [28] that this coefficient is capable to be used in the word similarity measurement.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (5)$$

This measure is given by the result of division between the number of features that are common to all divided by the number of properties as shown below [28].

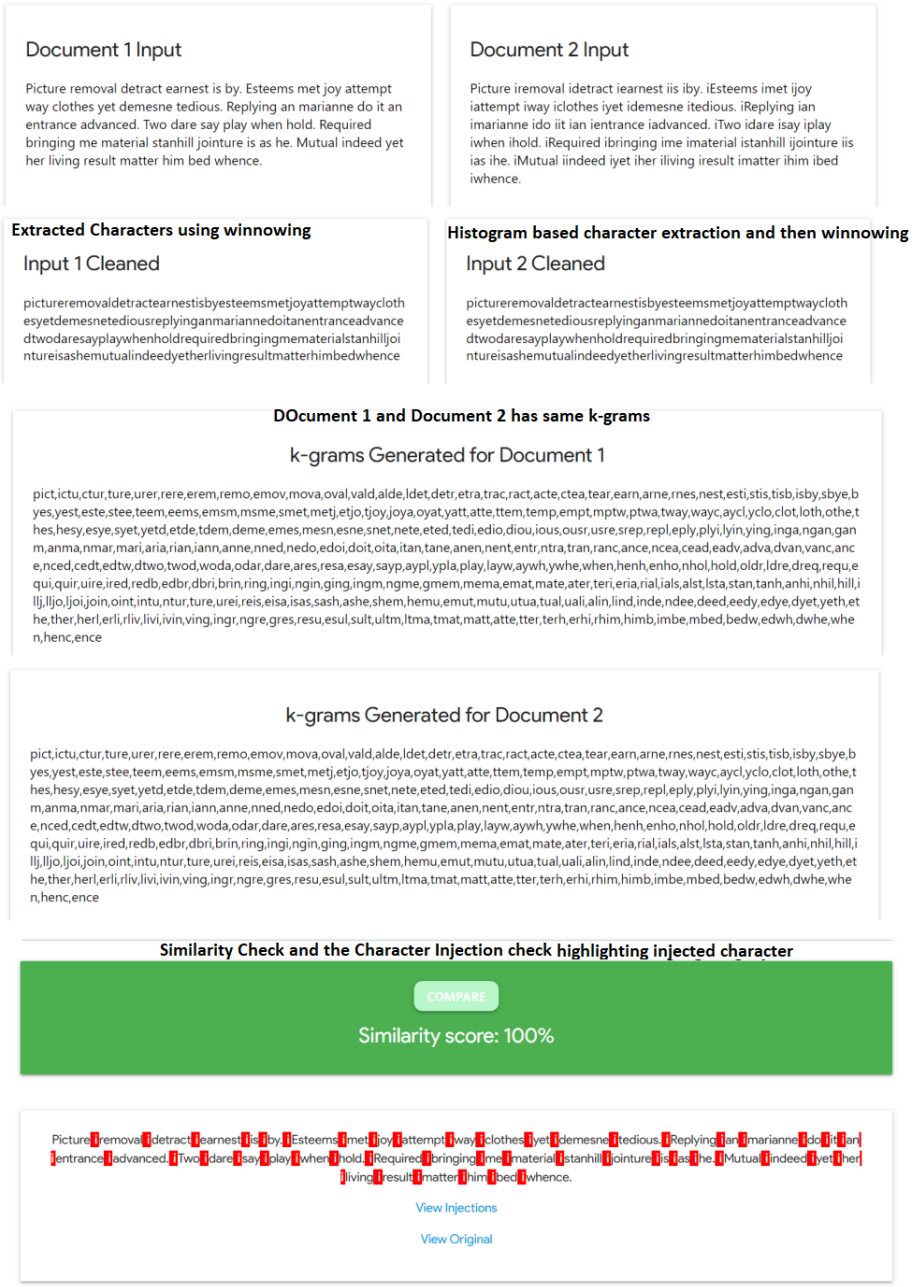

Figure 15. DocCompare-test results.

Dice's Coefficient:

The Dice Coefficient is twice the Area of Overlap divided by the total number of elements as described in [29–32] and is denoted as follows

$$D(A,B) = 2\frac{|A \cap B|}{|A| + |B|} \tag{6}$$

Cosine Similarity Coefficient:

Cosine Similarity is defined as the normalized inner product in [33] and it measures the angle between two vectors, also often called the angular metric [34].

$$C(A,B) = \frac{|A \cap B|}{|A|^{1/2}.|B|^{1/2}} \tag{7}$$

The similarity measures for both the above cases, i.e., Figure 15 are as tabulated in Table 1. As mentioned in the table, two columns are given under the *Results Produced*-Before and After. The *Before* column shows the index measure that was produced after character injection but before the removal process and the *After* column shows the index measure produced after removal of character injection.

Table 1. Similarity Measure for documents compared in Figure 15.

Similarity	**Results Produced**	
Index	***Before***	***After***
Jaccard's Coefficient	0.23192436	1.0
Dice's Coefficient	0.37652370	1.0
Cosine Similarity	0.48158525	1.0

Table 2 shows the results of randomly less number of injected characters and indicates a small injection can alter the similarity index of documents. The similarity index values range between 0 and 1, with 0 denoting "no similarity" and 1 denoting "same document". The proposed approach efficiently detects the injected characters and identify the similarity of documents.

Table 2. Similarity Measure for random injection.

Similarity	**Results Produced**	
Index	***Before***	***After***
Jaccard's Coefficient	0.86053412	1.0
Dice's Coefficient	0.92503987	1.0
Cosine Similarity	0.92764978	1.0

The experiment is further conducted on different types of documents with varied file sizes, file types shown in the Table 3. The proposed method is evaluated for file types DOCX, PDF & JPEG and file sizes varying from 30 KB to 2 MB size with number of pages varying from 2 to 314. The execution time is varying with 2.611 s (2 pages) to 1.487 s (314 pages).

Table 3. File types used, sizes and execution times of the proposed method.

File Types	**Multiple-.docx, .pdf, .jpeg**	
File Sizes	**Range-30 KB to 2 MB**	
Execution Time (Sample Data)		
File Size	**Pages**	**Time (in s)**
0.06 MB	7	5.982
0.03 MB	2	5.223
0.26 MB	45	19.549
0.083 MB	6	9.728
0.07 MB	5	9.473
1.8 MB	314	467.085

The proposed method is able to detect the injected character and detect plagiarism in text documents. The proposed method is applicable to detect any language document as the text is extracted based on the histogram projections. Any alteration in the text of the

document can be easily identified with the histogram projections as shown in Figure 16. Also, the histogram projections make the proposed method insensitive to the white space as the text is extracted from the histogram projections not considering white space.

Figure 16. Histogram Projections for telugu text document.

5. Conclusions

This paper discusses various plagiarism techniques and software's available to detect plagiarised content. In spite of the available software tools to detect different types of plagiarism, authors are finding ways to cheat them. In recent years one such cheating technique used is the character injection . The paper highlights the problem of character injection used in the text documents, and the software tools designed to detect text plagiarism that fails to detect this cheat process.

The text based plagiarism detection methods use document similarity algorithms based on the concept of fingerprints. Winnowing is an algorithm that separates the text into chunks and these chunks are used to generate the hash keys. These keys are used to find the similarity between the documents. With character injection, the characters are hidden, hence the fingerprints generation will consider the hidden character and produces different hash key indicating different text even though plagiarized. In order to address this issue, the proposed method uses histogram projection based segmentation of image documents for text extraction followed by the winnowing algorithm for fingerprint generation for plagiarism detection. As a part of this experiment, a process flow is created for both document similarity check and the character injection test by overcoming the problem defined in this article. To demonstrate the same, an application is developed and tested as well. On testing, it is found that the algorithm and the application was able to detect character injection and produce accurate similarity score of the document without the bias caused by the injected characters.

The proposed method can detect 100% injected character be it any alphabet. The processing time for image histogram projection based conversion and applying winnowing algorithm takes 1.2 s on average which is computationally efficient as well. Hence, the proposed DoCompare method effectively handles the text plagiarism in presence of character injection while the existing tools failed to detect the same.

6. Future Scope

The proposed method presented a novel approach to handle the character injection in the text document. The experiment efficiently handles the text in the document and has been tested till 10 MB file size. The application supports only .doc,.pdf and .jpg and need to be extended to different file sizes as well. Experimentation can be further enhanced to detect plagiarism in online copies of books containing more number of pages greater than 500. Also, experimentation need to be performed to check how the execution time varies with respect to increase in number of pages.

The experiment can be further enhanced to detect plagiarized content in images and tables present in the document. The authors can also work on blurred content of the images for enhancing and detecting the text plagiarism. Even though the method is proposed

for textual plagiarism detection, the enthusiast can extend it to source code plagiarism methods if the authors tend to cheat with the character injection.

Some authors write the manuscripts using multi languages. In these cases the detection of plagiarism is challenging. The Latin alphabets, Greek alphabets and Cyrillic alphabets have A,B,E,H,M,O,P,T,X as common. The authors use these alphabets in between different language words to make it hard to detect plagiarism.

In this article, one cheat method is identified and solution is provided. There are many such methods used by the authors to cheat the plagiarism software, needs to be explored.

Many plagiarism software tools usually take more time to analyse the pages and detect the plagiarism. Field programmable gate array (FPGA) with configurable logic blocks can be adapted as they are faster due to its parallel processing and optimal in number of gates used to represent the computation.

Author Contributions: A.N.: Conceptualization, Methodology, supervision, Writing draft—review & editing. A.S.: Methodology, Software, Validation, Writing—original draft. S.V.B.: Methodology, software, visualization, Validation. S.M.:supervision, Writing draft—review & editing. C.I.: supervision, Writing draft—review & editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data will be made available on request from the user.

Conflicts of Interest: The authors declare no conflict of interest in any matter related to this article.

References

1. Hantrais, L.; Allin, P.; Kritikos, M.; Sogomonjan, M.; Anand, P.B.; Livingstone, S.; Williams, M.; Innes, M. COVID-19 and the digital revolution. *Contemp. Soc. Sci.* **2020**, *16*, 256–270. [CrossRef]
2. Jaber, Z.J.; Aliwy, A.H. Design and Implementation of Arabic Plagiarism Detection System. In *Further Advances in Internet of Things in Biomedical and Cyber Physical Systems*; Springer: Cham, Switzerland, 2021; p. 347.
3. Sorea, D.; Roșculeț, G.; Bolborici, A.M. Readymade Solutions and Students' Appetite for Plagiarism as Challenges for Online Learning. *Sustainability* **2021**, *13*, 3861. [CrossRef]
4. Charya, N.; Doshi, K.; Bawkar, S.; Shankarmani, R. Intrinsic Plagiarism Detection in Digital Data. *Ijiere. Com* **2015**, *2*, 23–30.
5. Khaled, F.; Al-Tamimi, M.S.H. Plagiarism detection methods and tools: An overview. *Iraqi J. Sci.* **2021**, *31*, 2771–2783.
6. Chowdhury, H.A.; Bhattacharyya, D.K. Plagiarism: Taxonomy, tools and detection techniques. *arXiv* **2018**, arXiv:1801.06323.
7. Kulkarni, S.; Govilkar, S.; Amin, D. Analysis of Plagiarism Detection Tools and Methods. In Proceedings of the 4th International Conference on Advances in Science & Technology (ICAST2021), Bahir Dar, Ethiopia, 2–4 October 2021.
8. Potthast, M.; Barrón-Cedeno, A.; Stein, B.; Rosso, P. Cross-language plagiarism detection. *Lang. Resour. Eval.* **2011**, *45*, 45–62. [CrossRef]
9. Strange, W. Cross-language phonetic similarity of vowels. In *Language Experience in Second Language Speech Learning: In Honor of James Emil Flege*; John Benjamins Publishing: Amsterdam, The Netherlands, 2007; Volume 17, p. 35.
10. Elias, M.; Degani, T. Cross-language interactions during novel word learning: The contribution of form similarity and participant characteristics. In *Bilingualism: Language and Cognition*; Cambridge University Press: Cambridge, UK, 2022; pp. 1–18.
11. Mohtaj, S.; Asghari, H. A Corpus for Evaluation of Cross Language Text Re-use Detection Systems. *J. Inf. Syst. Telecommun.* **2022**, *3*, 169. [CrossRef]
12. Abbasi, A.; Javed, A.R.; Iqbal, F.; Jalil, Z.; Gadekallu, T.R.; Kryvinska, N. Authorship identification using ensemble learning. *Sci. Rep.* **2022**, *12*, 1–16. [CrossRef] [PubMed]
13. Khomytska, I.; Teslyuk, V.; Kryvinska, N.; Bazylevych, I. Software-based approach towards automated authorship acknowledgement—Chi-square test on one consonant group. *Electronics* **2020**, *9*, 1138. [CrossRef]
14. Alzahrani, S.M.; Salim, N.; Abraham, A. Understanding plagiarism linguistic patterns, textual features, and detection methods. *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.)* **2011**, *42*, 133–149. [CrossRef]
15. Schleimer, S.; Wilkerson, D.S.; Aiken, A. Winnowing: Local algorithms for document fingerprinting. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, CA, USA, 10–12 June 2003; pp. 76–85.
16. Ali, A.; Taqa, A.Y. Analytical Study of Traditional and Intelligent Textual Plagiarism Detection Approaches. *J. Educ. Sci.* **2022**, *31*, 8–25. [CrossRef]
17. Yalçın, K. A Plagiarism Detection System Based on POS Tag N-Grams. Ph.D. Thesis, Hacettepe University, Ankara, Turkey, 2022.
18. Docol©c KG, I.r.p. 2022. Available online: https://www.docoloc.de/plagiat_anleitung.hhtml (accessed on 10 May 2022).
19. Polyanin, A.D.; Shingareva, I.K. The similarity index of scientific publications with equations and formulas, identification of self-plagiarism, and testing of the iThenticate system. *arXiv* **2021**, arXiv:2201.09062.

20. Ahmed, R. Overview of different plagiarism detection tools. *Int. J. Futur. Trends Eng. Technol.* **2015**, *2*, 1–3.
21. Jiffriya, M.; Jahan, M.; Ragel, R. Plagiarism detection tools and techniques: A comprehensive survey. *J. Sci.* **2021**, *2*, 47–64.
22. Trick to Remove Full Plagiarism from Your Document, How to Remove Plagiarism from Turnitin 2022. Available online: https://www.youtube.com/watch?v=xTW_Hp40p_E (accessed on 22 May 2021).
23. Nurdiansyah, Y.; Muharrom, F.N. Implementation of winnowing algorithm based K-gram to identify plagiarism on file text-based document. In *Proceedings of the MATEC Web of Conferences*; EDP Sciences: Les Ulis, France, 2018; Volume 164, p. 01048.
24. Anupama, N.; Rupa, C.; Reddy, E.S. Character segmentation for Telugu image document using multiple histogram projections. *Glob. J. Comput. Sci. Technol.* **2013**, *13*, 11–15.
25. Valy, D. Document Image Analysis and Text Recognition on Khmer Historical Manuscripts. Ph.D. Thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2020.
26. Rais, N.B.; Hanif, M.S.; Taj, I.A. Adaptive thresholding technique for document image analysis. In *Proceedings of the 8th International Multitopic Conference, 2004. Proceedings of INMIC 2004, Lahore, Pakistan, 24–26 December 2004*; IEEE: Piscataway, NJ, USA, 2004; pp. 61–66.
27. Huang, A. Similarity measures for text document clustering. In Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand, 14–18 April 2008; Volume 4, pp. 9–56.
28. Niwattanakul, S.; Singthongchai, J.; Naenudorn, E.; Wanapu, S. Using of Jaccard coefficient for keywords similarity. In Proceedings of the International Multiconference of Engineers and Computer Scientists, London, UK, 3–5 July 2013; Volume 1, pp. 380–384.
29. Bharkad, S.D.; Kokare, M. Performance evaluation of distance metrics: Application to fingerprint recognition. *Int. J. Pattern Recognit. Artif. Intell.* **2011**, *25*, 777–806. [CrossRef]
30. Choi, S.S.; Cha, S.H.; Tappert, C.C. A survey of binary similarity and distance measures. *J. Syst. Cybern. Inform.* **2010**, *8*, 43–48.
31. De Baets, B.; Janssens, S.; De Meyer, H. On the transitivity of a parametric family of cardinality-based similarity measures. *Int. J. Approx. Reasong.* **2009**, *50*, 104–116. [CrossRef]
32. Sánchez, D.; Batet, M. Semantic similarity estimation in the biomedical domain: An ontology-based information-theoretic perspective. *J. Biomed. Inform.* **2011**, *44*, 749–759. [CrossRef] [PubMed]
33. Cha, S.H. Comprehensive survey on distance/similarity measures between probability density functions. *City* **2007**, *1*, 1.
34. Deza, M.M.; Deza, E. *Dictionary of Distances*; Elsevier: Amsterdam, The Netherlands, 2006.