USING OCR FOR DOCUMENT INTEGRITY CHECKING

Submitted by - Sankari Balasubramaniyan,
under the supervision of Dr. Pauline Puteaux and Dr. Iuliia Tkachenko

February 2023

1 INTRODUCTION

The main objective of this project is to study the issue of effective use of OCR and fuzzy extractors to determine the falsification of printed and scanned documents. To comprehend the functionalities of this project few key developments have been performed :

1. Bibliographic study to take inspiration from existing methodologies,
2. Automatic identification of areas of interest in the document using Region of Interest (ROI) approach,
3. Pre-process the ROI, use of the Tesseract OCR method for text recognition and post process the detected regions to reduce common errors from OCR,
4. Applied several word distance algorithms - Levenshtein, Jaccard, Hamming, a custom word distance code and ssdeep , pypi fuzzy extractors to identify falsification and
5. Evaluated the performance on new datasets, along with highlighting the regions on the table to the users of potential falsification.

2 SYSTEM REQUIREMENT

- Python3
- Pytessarct OCR
- Fuzzy extractor pypi
- Ssdeep
- OpenCV
- Skimage

3 METHODOLOGY

3.1 DATASET PRE-PROCESSING IMPLEMENTATION

1. **Binarization methods using Otsu**

   Thresholding is an image segmentation process for converting a grayscale image to a binary image with only black and white pixels. This process can be used to reduce the emphasis on background pixels and more on the foreground containing the text, this allows us to select areas of interest of the document while ignoring the parts that are not concerning. There are numerous different methods to perform thresholding method, for the purpose of this project Otsu methodology is used which is a technique that examines all potential values for the threshold separating the background from the foreground, calculates the variance within each of the two clusters, and chooses the value for which the weighted sum of these variances is the smallest. In this project its mainly used as a preprocessing method to target text-based regions.

2. **Noise Reduction methods**

   - **Gaussian filter**

     OpenCV's inbuilt Gaussian filter function is used to remove gaussian noise, which is the type of noise predominantly present in printed and scanned or mobile-captured document images. Gaussian filter is a 2D convolution operation which is used to remove the said noise from image. Gaussian filtering is done by convolution of each pixel in the input image with a gaussian kernel (whose size is set to (5,5))and then summing them all to produce the output image. In this project, it is mainly used to blur the small unwanted pixels persisting in the document due to PS operations.

   - **Non-local means**

     The Non-Local Means (NLM) algorithm replaces a pixel's value with the average of a subset of its nearby pixel values. The patch centering on the pixel of interest is compared to small patches centered on the other pixels, and the average is only calculated for pixels with patches that are close to the current patch. An implementation of this is provided using opencv's function fastNlMeansDenoising. Since this method takes a mean value of all pixels to smooth the image, it is possible that this algorithm can restore small textures that can be regarded unnecessary for this project.

3. **Image Sharpening methods - Unsharp Mask**

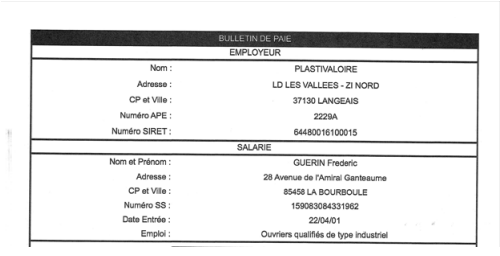
   This is the process of sharpening an image or performing edge enhancement using a smoothing filter. We first blur the image. Smoothing an image will largely conceal its high-frequency components which are performed using gaussian filter. Then, the smoothed image is subtracted

from the original image (the resulting difference is the required mask). Now, the output image will have most of the high-frequency components that are blocked by the smoothing filter. Then this mask is added back to the original image which results in enhancing the high-frequency components. But in our case, it might introduce more noise and discernable edge effects.

4. **Skew Transformation**

   Deskewing has been a mandatory preprocessing step for some printed and scanned images which has gone through some geometrical transformation, before feeding it to the OCR. The main goal is to not affect the text (separate it from the background) and determine the angle of transformation. Initially, the image is loaded as grayscale and blurring process is applied to reduce noise present in the image. Thresholding method based on Otsu is applied which inverts the image pixels and binarizes them (making the texts as white and the background black which eventually makes larger portion of the image black). Now in order to find the text block, the white pixels width are increased drastically using dilation process (which merges the characters indistinguishably) with rectangle as the structuring element that has size (30, 5), where the X axis gets rid of all spaces between words, and a smaller kernel on Y axis to blends lines of one block between each other, but keep larger spaces between text blocks intact. Now contours function is applied to find all the joined objects of the image (which signifies different text blocks), and the largest contour is chosen to determine skew angle. Once the angle calculation is done, we just need to re-rotate the image around its centre. Not all the documents require skew transformation, for instance in the Dataset_1 only genuine 2PS600 PaySlip_Arial_10_1g, forged 2PS600 Imitation_1_PaySlip_Arial_10_1f_1 and forged PS600 Imitation_1_PaySlip_Arial_10_1-f_1 needed transformation. If the skew correction is not performed, it leads to detection of random digits and characters that are not originally present in the document by OCR.

   Skewed


BULLETIN DE PAIE

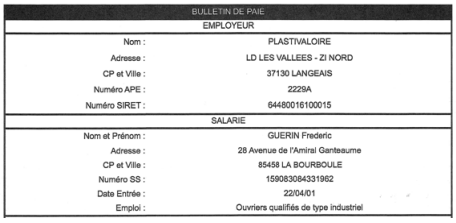
EMPLOYEUR

Nom :	PLASTIVALOIRE
Adresse :	LD LES VALLEES - ZI NORD
CP et Ville :	37130 LANGEAIS
Numéro APE :	2229A
Numéro SIRET :	64480016100015

SALARIE

Nom et Prénom :	GUERIN Frederic
Adresse :	28 Avenue de l'Amiral Ganteaume
CP et Ville :	85458 LA BOURBOULE
Numéro SS :	159083084331962
Date Entrée :	22/04/01
Emploi :	Ouvriers qualifiés de type industriel

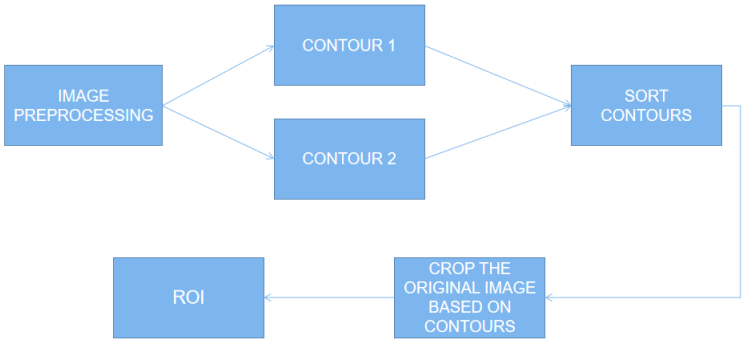
   Deskewed



BULLETIN DE PAIE

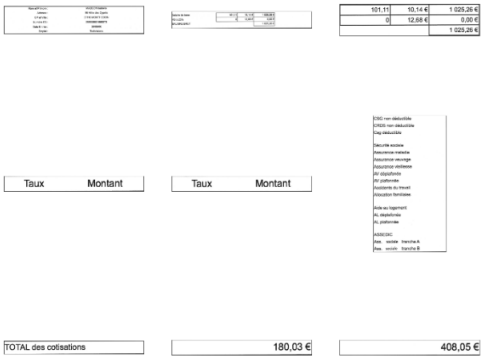
EMPLOYEUR	
Nom :	PLASTIVALOIRE
Adresse :	LD LES VALLEES - ZI NORD
CP et Ville :	37130 LANGEAIS
Numéro APE :	2229A
Numéro SIRET :	64480016100015
SALARIE	
Nom et Prénom :	GUERIN Frederic
Adresse :	28 Avenue de l'Amiral Ganteaume
CP et Ville :	85458 LA BOURBOULE
Numéro SS :	159083084331962
Date Entrée :	22/04/01
Emploi :	Ouvriers qualifiés de type industriel

3.2 REGION OF INTEREST



Once preprocessing algorithms are applied to the image, it is now time to find our areas of interest - which are only the regions containing texts. The pipeline to find ROI consist of three important operations - determine contours of the image, sort contours based on their position and crop the image based on chosen contours to obtain the ROI. Two types of contour functions are implemented based on the size of the original image. Initially, a thresholding function is applied to the preprocessed image and some morphological functions are applied to increase the width of the pixels of outer contour of the image (so it will be more prominent) and remove white noise by first applying dilation and then erosion with small rectangle structuring elements. Then all the contours of the images are found and sorted from top to bottom and left to right based a contour sorting algorithm. This algorithm works as such, initially, all the contours are sorted from top to bottom. Now the contours in the same y-axis are sorted based on ascending x-axis values. This algorithm will ensure we have the same order of contours irrespective of the chosen image. Now the contours are filtered based on their size, this is part of contour algorithm which is different for different resolution images due to the reason that size of the image directly impacts the area of the contour. The area threshold is chosen by manually analyzing every contour size and this allows us to only have the contours of the image which contribute to a particular table in the given document.

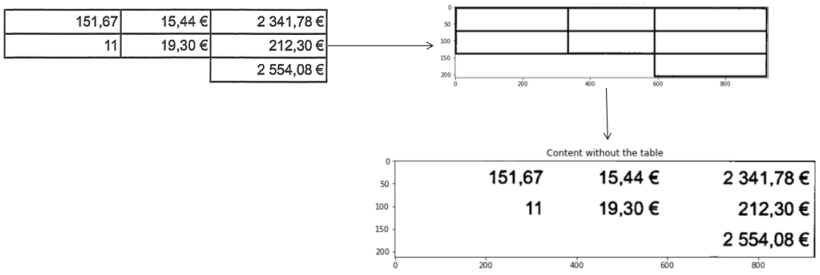
Example of result



3.3 IMAGE PREPARATION FOR OCR

Once the ROI is found, we are back to dealing with the original images that went through preprocessing, only difference is that they are cropped to target specific locations of the image. This point is emphasized due to the fact that before feeding the OCR some modifications have to be made to accurately recognize texts present in the image. For this purpose, some preparation is performed on the obtained ROI. The accuracy of OCR technology is determined by the quality of the letters, hence importance is given to sharpening character borders, improving contrasts, well-orienting characters, and reducing pixel noise as much as possible. The first step is to scale the resolution of the image, this is done because it's been proved that when OCR scans the image pixel by pixel, it reads the image with the highest character dimensions above 20 pixels with high accuracy. After this, opening and closing morphological operations are applied. Opening and closing are operators that are derived from the fundamental operations of erosion and dilation and are normally applied to binary images to remove white noise and close the structure of the edges of the image. Opening tends to use erosion to thin some edges present in the foreground of the image introducing space between two close characters, detaching objects that are joint and should not be, and enlarging holes inside objects while thinning unwanted small pixel noises present in the image. Closing operation uses dilation to enlarge the boundaries of foreground regions in an image while joining broken edges and fill in unwanted holes in the object.

Subsequently, it was noticed that some error of OCR was contributed due to the presence of tables in the ROI (which are read as character 'I' by OCR). Those tables are removed from the image leaving just the content of the table (i.e, texts). This is achieved using a different algorithm by creating a mask of all the contours contributing to the table of the image and then subtracting the mask from the original image.



Finally, there have been some images that contain ROI inside an ROI and those are effectively removed to avoid redundancy in the result.

3.4 OCR DETECTION

OCR Pytessaract version v0.3.10 is used for the purpose of this project. The ROI are preprocessed before sending to the OCR. For digits, the configuration is set to detect only the digits present in the given image, it is observed that enabling this does not change the result (detection remains the same) and for text, the images can be fed into the system without any configurational changes. The result is sent to postprocessing to remove and reduce redundant errors that can be eliminated without affecting the integrity of falsification detection.

3.5 POST PROCESSING IMAGE FROM OCR

The stability of OCR output is quite important since wrong OCR detection might lead to wrong document fraud detection. In order to reduce some disambiguate and common errors from OCR, a set of post-processing techniques has been used which will be described in this section. The stability of OCR in our case means that the sequence of characters produced by the OCR algorithm should always be the same for similar inputs, for instance even if there's been an error in OCR detection if that error persists in every output, we can successfully ignore it. We can consider for our case that the accuracy of OCR is not as important as its stability. Stated below are some of the changes made to OCR results.

- Redundant ' \n' is removed, as it seems unnecessary to have an extra line.

- All the letters are converted to lowercase, this is performed in an attempt to check if the accuracy of the system improves but OCR detection was quite accurate for distinguishing lower and uppercase

- Unnecessary characters such as

$\sim, -, \_, ., |, (,), [,], \{, \}, , , ; , *, ', \ll, , =, <, >, \#, |$

removed, as these are irrelevant to the results and also not commonly found in the documents.

- Alphabet reduction is a technique which is introduced to remove ambiguity in OCR because sometimes its difficult for the system to distinguish an I from

an l, hence removing and replacing some of such characters does not affect the readability of the document but improves the OCR stability. Example - '&' to 'a' and 'é' to 'e' and 'l' or 'i' to '—' and 'ee' to 'e'

- To make the outputs more predictable or similar, a constant order of sentences is maintained irrespective of resolutions of ROI.

- Remove leading and ending whitespace.

3.6 FALSIFICATION DETECTION METHODS

1. **ssdeep**

   For this project, a python wrapper of ssdeep library built based on computing context-triggered piecewise hashes (CTPH) is used to determine falsification in the document. This type of fuzzy hash is mostly used for homologous data which shares identical sets of bits in the same order. CTPH is a method for creating hash signatures that combines a number of conventional hashes whose boundaries are determined by the context of the input. These signatures can then be used to identify falsified versions of known documents even if data has been inserted, modified, or deleted. CTPH uses the rolling hash which is a recursive algorithm that produces pseudo-random values based on the fed input and is hashed in a window that moves through the input. While processing the input, both the rolling hash and piecewise hash is computed simultaneously. Piecewise hash is a technique used to segment every fixed length interval of data into pieces, calculate the hash value for each piece, and compare these hash values together for similarity. The value of the traditional hash is registered in the CTPH signature when the rolling hash generates a trigger value, and the traditional hash is then reset. As a result, each recorded value in the CTPH signature only depends on a small portion of the input, and modifications to the input will only cause small modifications to the CTPH signature. For instance, if a single byte of the input is altered, just one of the conventional hash values will be altered; the majority of the CTPH signature will stay unchanged. Documents with alterations can still be connected to the original document with CTPH signatures because the majority of the signature is still the same. This is one of the reasons why fuzzy extractors such as ssdeep can be used efficiently in our project, unlike SHA-256 which will produce an entirely different signature for even really small falsifications.

   Finally, a matching score of two fuzzy hash values is computed by the weighted average distance. Their in-built function 'compare' takes two generated hashes as the input and provides a similarity measure, where 100 means the hashes are exactly the same and close to 0 signifies major changes made to the original document. This weighted average distance is calculated by determining the number of operations required to change from signature1 to signature2 (including insert, delete, modify, and exchange), and then give a weight to different operations. The sum of the

results gives us the weighted edit distance. Then divide this distance by the sum of the lengths of signature1 and signature2 and make the absolute result a relative result that will help mapping it to an integer value of 0-100.

Observations

Example 1

```
---------------------------------------------------------


Output  1

nom: adresse: cpetv||e: numeroape: numeros|ret: t|groupautomot|vesystems bdde|'|ndustr|e 37530naze||esnegron 24202 54210000300053

3:vFFNLUK8AtLUKzfrZk5sIKkWcWFBA8ZLCnnAMaxDXqmMD:vFFZdtrNk5sIKfnE8ZpFxWmMD
3:vFFNLUK8AtLUKzf9wIWnnAMaxDXqmMD:vFFZdtrioFxWmMD

nom: adresse: cpetv||e: numeroape: numeros|ret: t|groupautomot|vesystems bdde|'|ndustr|e 37530naze||esnegron 24202 54210000300053
nom: adresse: cpetv||e: numeroape: numeros|ret: t|groupautomot|vesystems bdde|'|ndustr|e 37530naze||esnegron 24202 54210000300053
nom: adresse: cpetv||e: numeroape: numeros|ret: t|groupautomot|vesystems bdde|'|ndustr|e 37530naze||esnegron 24202 54210000300053
nom: adresse: cpetv||e: numeroape: numeros|ret: t|groupautomot|vesystems bdde|'|ndustr|e 37530naze||esnegron 24202 54210000300053
nom: adresse: cpetv||e: numeroape: numeros|ret: t|groupautomot|vesystems rtede|'|ndustr|e 37530naze||esnegron 24202 54210000300053
nom: adresse: cpetv||e: numeroape: numeros|ret: bdde|'|ndustr|e 37530naze||esnegron 24202 54210000300053
100 100 100 100 46 31
---------------------------------------------------------
```

- Shown above is an example of a result from ssdeep, as we can see it can able to identify the falsification by providing a score of similarity instead of completely generating a different signature which would have denied us from analyzing the effect of falsification. In the figure the signatures ”3:vFFNLUK8AtLUKzfrZk5sIKkWcWFBA8ZLCnnAMaxDXqmMD:vFFZdtrNk5sIKfnE8ZpFxWmMD” and ”3:vFFNLUK8AtLUKzf9wIWnnAMaxDXqmMD:vFFZdtrioFxWmMD” correspond to the last two falsified strings whose genuine signature is ”3:vFFNLUK8AtLUKzfrZk5sIKkWcWXaWnnAMaxDXqmMD: vFFZdtrNk5sIKfnXeFxWmMD”. We can observe that the falsification of the document has impacted the signatures to change in the middle of first and second of signatures. This lead to score to be in the range 30 to 50 which also depends on the length of the sentences.

Example 2

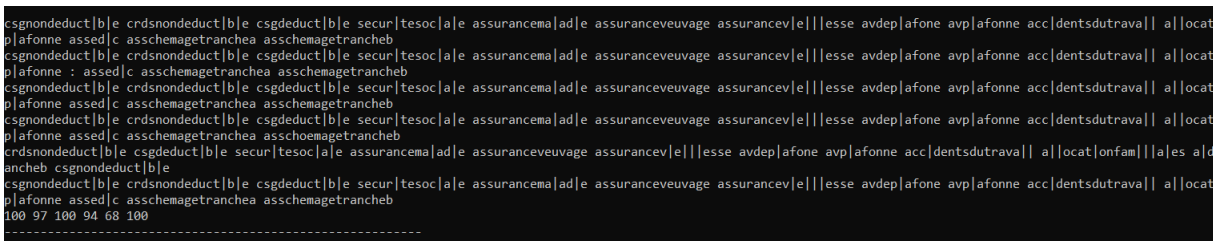
```
Output  3

nometprenom: adresse: cpetv||e: numeross: datentre: emp|o|: masson|sabe||e 96a||edescypres 37110monthodon 269038631660073 30/09/06 techn|c|ens

3:wzXFUNLUKe8dMgAmQJXUikaKLtVWo4CTfZUALW:cVUZidtkB6otZi
3:wzXFhmMLUKe8dMgAmQJXUikaKLtVWo4CTfZUALW:cVFidtkB6otZi

nometprenom: adresse: cpetv||e: numeross: datentre: emp|o|: masson|sabe||e 96a||edescypres 37110monthodon 269038631660073 30/09/06 techn|c|ens
nometprenom: acresse: cpetv||e: numeross: datentre: emp|o|: masson|sabe||e 96a||edescypres 37110monthodon 269038631660073 30/09/06 techn|c|ens
nometprenom: adresse: cpetv||e: numeross: datentre: emp|o|: masson|sabe||e 96a||edescypres 37110monthodon 269038631660073 30/09/06 techn|c|ens
nometprenom: adresse: cpetv||e: numeross: datentre: emp|o|: masson|sabe||e 96a||edescypres 37110monthodon 269038631660073 30/09/06 techn|c|ens
nometprenom: adresse: cpetv||e: numeross: datentre: emp|o|: masson|sabe||e 96a||edescypres 37110monthodon 269038631660073 30/09/06 techn|c|ens
nometprenom: adresse: cpetv||e: numeross: datentre: emp|o|: masson|sabe||e 96a||edescypres 37110monthodon 269038631660073 30/09/06 techn|c|ens
100 38 100 100 100 100
---------------------------------------------------------
```

In this case, the error is in the spelling of ”address” in the second document which is caused due to OCR. Though there is only a single misplaced

character, it can be observed that (considering the first signature as the original and the second as the falsified), there are changes in the first and second signatures in the middle and hence we still get a score between 30 to 50. The result remains the same irrespective of the location of error (in the beginning or end) and irrespective of whether the error is due to addition or deletion or replacement.

Also, it is important to notice here that the score more or less remains the same, whether there has been an addition of one or three characters (it can be seen in the figure below, I introduced this error for the purpose of analysis).

Example 3



However, in this example it can be noted that there is an addition of ':' in the second data and 'o' in chemage of fourth data. There is an addition of a character just like the previous example, the scores here are more than 90 for both cases. Hence it can be observed that the same type of error leads to different scores, and the value of score is impacted based on the length of input sequences. An error in a small sequence reduces the score quite a lot while the same error in a large sequence is almost negligible.

Some of the drawbacks of using ssdeep are:

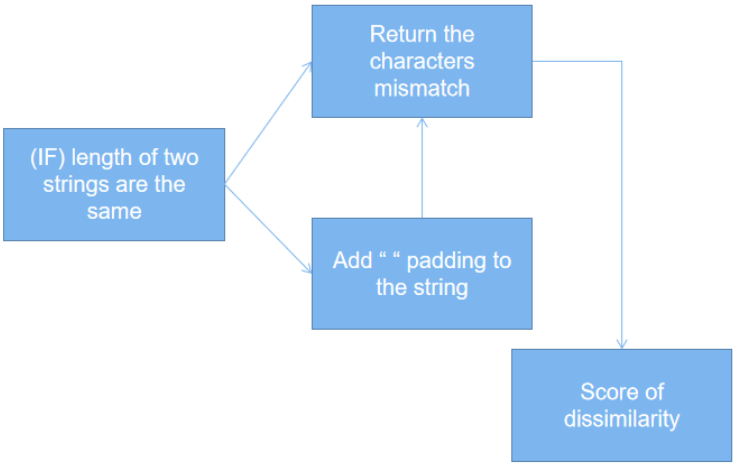
- Data of varying lengths can have equal-length hash values due to fixed length piecewise hashing algorithm, however, when comparing data of different lengths, alignment issues occur. Also, it is mentioned by the developer of the algorithm that, when implementing longer data comparisons,

it is necessary to process adjustments on the segmentations several times, which eventually deteriorates the accuracy of comparison.

- When the size of input data is small, the algorithm is susceptible to unrelated characters and content (example whitespace).

2. **Hamming distance**

Hamming distance is one of the most popular metric used to compare two strings to check if they are equal. It is measured as the number of character positions in which the characters are different. The function is defined such that, if the string lengths are equal, each characters of two strings are compared, and if the lengths are not equal then a space padding is added because hamming distance algorithm requires two string lengths to be the same. Hence the defined function will returns the number of characters that are not equal at every position in two given strings.



Observations

- Requires strings to have equal length.

- Can not detect pattern, i.e if there has been a shift in the characters due to addition or deletion of a character, the algorithm can not distinguish this and the score turns out to be low even if the impact is due to a single character.

3. **Jaccard similarity**

The Jaccard similarity is a statistical distance that measures the diversity and similarity of sample sets. It can be described by the formula given below where A and B are sets and J(A,B) determines the Jaccard similarity between those two sets. Here, the result is such that $0 \leq J(A, B) \leq 1$, if

A intersection B is empty, then J(A,B) = 0 and if A and B are the same, then J(A,B) = 1.

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

In our case, the original strings are converted to sets of words and calculated the size of the intersection divided by the size of the union of the two words sets.

Observations

- There is no localization of the error, because jaccard similarity we will not be able to identify the location of error in the string and also different-sized sets with same number of common members also will result in the same Jaccard similarity.

- However it solves the previous issue of not being able to determine pattern and the score should be able to tell the depth of falsification i.e if there's just one or two characters that are misplaced or if the entire sentence is changed.

Result

Accuracy					
Images	TruePositive	TrueNegative	FalsePositive	FalseNegative	Accuracy
1 (PS600 gen)	18	0	0	3	85.7
2 (PS300 gen)	19	0	0	2	90.48
3 (PS600 gen)	20	0	0	1	95.23
4 (PS600 forged)	18	1	0	2	90.48
5 (PS600 forged)	15	2	0	4	80.95
6 (PS300 forged)	18	1	0	2	90.48
7 (PS300 forged)	16	3	0	2	90.48
8 (PS600 forged)	17	1	0	3	85.7
9 (PS600 forged)	16	2	0	3	90.48

4. **Custom Fuzzy extractor**

The fuzzy extractor method can be used to determine the similarity score between two strings by measuring the difference between their fingerprints.

The code defines a function fuzzy_extractor_similarity(s1, s2, k=16) that takes two strings as input and returns a similarity score between 1 and 100, where 100 indicates that the strings are exactly the same and 1 indicates that the strings are completely different. The function uses the hashlib library to hash each string using the SHA-256 hash function, and then takes the first k characters of the hexadecimal representation of the hash to create a fingerprint. The k parameter is a security parameter that controls the length of the fingerprint and the likelihood of a false positive (i.e., the likelihood that two different inputs will have the same fingerprint). The function calculates the similarity score by measuring the Hamming distance between the two fingerprints, which is the number of positions in the two fingerprints that are different. The similarity score is calculated as 100 * (1 - (distance / k)), where distance is the Hamming distance and k is the length of the fingerprints.

Observations

- Even small changes lead to a huge differences in the signatures which makes it difficult to analyze if there is just a impact of falsification, if the error is due to a single character misplacement or entire sentence. Since, unlike the previous hashing technique (ssdeep), hash is not generated character by character, hence sometimes when the data length is too small, a minor error (like the insertion of a character in a sentence) makes the sentence as a different data from the original and the score turns out to be 0.

5. **Fuzzy extractor library**

   Concept - Two strings with minimal noise added will produce the same key if the noise is less than a threshold value. Hashes of two strings are determined using SHA256, and the reference string is hashed using generate an inbuilt function which in turn returns two values - key(secret) and helper(public) The modified string along with the helper is then used by the reproduce function to reproduce the key. This method also employs the hamming distance as its error measure, which means that if two binary strings' hamming distances are equal to or less than some predetermined threshold, they will almost certainly produce the same key.

   Observations

   - Does not work with '€' symbol

   - Strings of unequal length can not be directly used

   - Not robust to slight linear displacement

6. **Levenshtein distance**

   The Levenshtein distance is a string metric for measuring the difference between two sequences. The difference between two sequences is measured based on how many minimum numbers of single-word operations such as insertion, deletion or replacement are required to convert one sequence

to another. Matrix method is used for determining the distance which is coded using dynamic programming to gain better efficiency in time and space constraints. An example to convert "Hello Here" and "Hey There" is given below to demonstrate the functionality of the algorithm and construction of DP table.

- The words "Hello Here" and "Hey There" are written in columns and rows considering each of the words as a separate element of the matrix table.

- The first row and column are marked from 1 to the length of the sequence for each word in the table. 0 represents a null word meaning it has no alphabet or letters in it. It reflects the number of edits required to convert the word to the null string. So to convert a null string to "Hello Here" or "Hey There" it will require two edits (two insertions), "Hello" to "Hey" one edit, and "Hello here" to "Hey there" two edits.

- So the other values of the matrix are filled depending on how many edits are required to convert each word of the sequence of another or by the surrounding values. So an element is filled by taking the minimum of (diagonal, top and left) values and add 1. The diagonal, top and left represent operations replace, delete and insert, we take the minimum to consider the least number of operations until the previous step and add one to include the current operation for transformation.

- Finally, the last value in the DP table is the total number of Levenshtein distance required either to convert one sequence to another or in our case the prominent amount of falsification in the current data in comparison to the reference data.

	" "	Hey	There
" "	0	1	2
Hello	1	0	0
Here	2	0	0

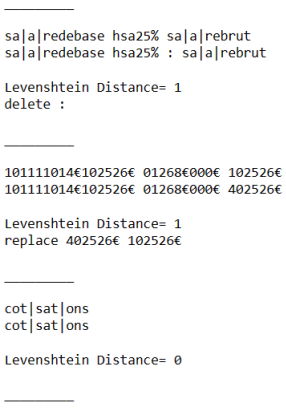
	" "	Hey	There
" "	0	1	2
Hello	1	1	2
Here	2	2	0

	" "	Hey	There
" "	0	1	2
Hello	1	1	2
Here	2	2	2

7. **Custom word distance**

This is an extended version of the Levenstein distance. Through the previously constructed DP table we should be able to localize the falsification and help the user correct it. This algorithm is developed going backward in the DP table by determining which operation is performed that led to the current Levenshtein distance. By the end, it will notify the user which operation (Replace, insert or deletion) is needed to be performed to remove falsification from the document and enhances readability since it can be quite exhausting the find the error after knowing there is one.

Example of result



4 RESULT AND DISCUSSION

This project attempts to solve the problem of falsification of important printed and scanned documents by providing a set of functions to develop local level tamper-proof system using fuzzy extractors and word distances algorithms. Here, we explored the usage of different fuzzy extractor implementations (such as ssdeep and fuzzy extractors based on SHA-256 hashing) and word distances (such as Hamming, Jaccard, Levenshtein) and also provided an implementation of a custom word distance algorithm to assist users with the type of operations required (addition, deletion or replacement) to revert back the original data from falsification. Accuracy is used as the evaluation metric on Dataset_2, for word distance algorithms (provided in Jaccard similarity section) which remains the same (though the score differs) for other distances as well because the algorithms were able to detect falsification 100%, leaving OCR detection as the only source of error that led to false negatives. Finally, a short algorithm is written to highlight the local regions of the image which could be potentially tampered (based on the results from the system) to enhance the visualization and readability when the document is being cross-examined by human operators. However, one of the limitations of the system is that (though we did not test) there can be a case that rises false positive if in a ROI, a value is removed

and placed in a different location. Since we read the data using OCR not considering the location of the data, this falsification is something that we can not detect.

Due to time constraints, the fuzzy extractor library was not analyzed to the fullest, so it will be a good option in the future to explore this and other different fuzzy hashes such as Block-Based Hashing (BHB) like dcfldd, Statistically-Improbable Features (SIF) of which the most popular one is sdhash and Block-Based Rebuilding (BBR) like mvHash-B.