A study of the factors influencing OCR stability for hybrid security

Sébastien Eskenazi, Petra Gomez-Krämer, Jean-Marc Ogier

A study of the factors influencing OCR stability for hybrid security

Sébastien Eskenazi, Petra Gomez-Krämer, and Jean-Marc Ogier
L3i, University of La Rochelle, Avenue Michel Crépeau, 17042, La Rochelle, France
Email: {sebastien.eskenazi, petra.gomez, jean-marc.ogier}@univ-lr.fr

***Abstract*—Optical character recognition (OCR) is a critical task in securing hybrid (digital and paper) documents. For this, its key performance criterion is stability. An unstable OCR algorithm will fail to detect two copies of the document as similar thus creating a wrong fraud detection. Having a sufficiently stable algorithm requires a very high level of performance. To improve it, we study a simple disambiguation technique called "alphabet reduction". It is based on the principle that characters that are visually similar should be the same character. It significantly improves the stability of two state of the art OCR algorithms on almost forty three thousand images. Yet the obtained stability is still insufficient. We also study the impact of the document variations on the stability of OCR algorithms.**

I. Introduction

With the ever increasing digitization of our world, people and companies now have to ensure the authenticity of many documents. For instance, an easy way to get a fraudulent identity card is not to forge one but to obtain a real one with fraudulent documents such as a fake electricity bill and a fake birth certificate [1]. These documents frequently change between a paper and a digital format. Ensuring the security of these documents even if they change format is called hybrid security.

Figure 1 shows a typical hybrid security system. The input documents (digital or images of paper documents) go through a content extraction step followed by a signature computation. The obtained signatures can then be compared to see if the documents are similar. Usually, one computes the signature of an alleged authentic copy and compares it with a secure signature of the original/authentic document. It is not necessary to have the authentic document, only its signature is needed. OCR algorithms are used in the content extraction from the images of the paper documents.

In the wake of the content-based signature for hybrid security proposed in [2], this paper provides an in depth analysis of the suitability of OCR algorithms for such security applications. We formalize the issue of the stability of an algorithm and we perform a more in depth study of the alphabet reduction. We also add a study of the impact of the document variations on the stability of OCR algorithms.

This paper is organized as follows. We present the problem and the state of the art of OCR algorithms. Then we formalize the definition of a stable algorithm. We continue with the presentation of the alphabet reduction and its evaluation before concluding this paper.

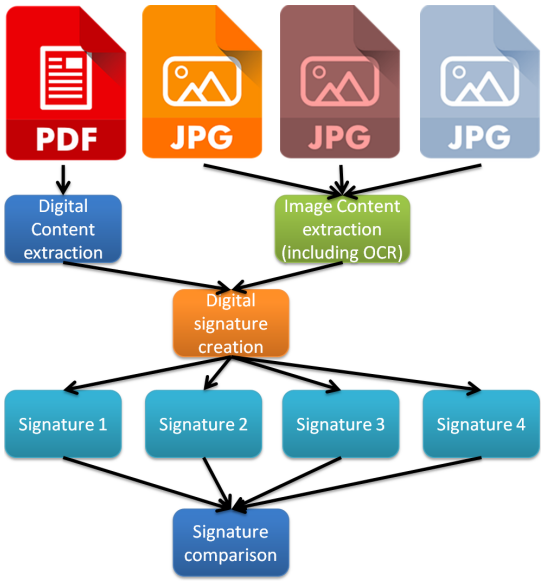

Fig. 1: A generic hybrid document securing and verification process

II. Problem statement

In this paper, we focus on recognizing printed text because the technology is not mature enough for handwritten text [3]. Printed Latin modern text recognition is considered by many as a solved problem and most software frequently reach character error rates below 5-10%. However our main performance criterion is the stability of the algorithm, which has barely been studied. The stability of an OCR algorithm means that the sequence of characters produced by the OCR algorithm should always be the same for similar inputs. In our case, similiar inputs are the authentic document and images of the authentic document. Hence, either the OCR algorithm does not make errors on copies of the same document or it makes the same errors on all copies. On Figure 1, if the OCR fails to produce the same output on all the copies of the same paper document, some will be wrongly declared as modified.

A page contains approximately 2000 characters. A 5 % false negative rate means that one page can be different out of 20 similar pages. If we assume that the OCR does not make any mistake on the other 19 pages, this implies a character error rate below 1 out of 38 000 or 0.0026 %. It is also possible that the OCR makes some mistakes, in which case it should

make the same mistakes on the 19 pages.

Besides this extremely low character error rate, the OCR algorithm should be able to use either text lines or unsegmented document images as inputs. This is due to the fact that document image segmentation algorithms can only extract text lines or text blocks. Furthermore the OCR algorithm should not rely too heavily on a word dictionary since many administrative documents contain names and item references. The other issue with dictionaries, is that the algorithm output should remain true to the original text including its errors. A dictionary based approach brings the risk of correcting the errors contained in the document. This issue is even more severe when dealing with fraud detection. Any kind of correction could actually hide the fraudulent modification and this is not acceptable.

We will now present the state of the art OCR techniques.

III. State of the art

OCR algorithms for type-written text can be divided in two main trends: creating a new OCR algorithm and improving an existing OCR without modifying it, for instance by adding a pre- or post-processing. The alphabet reduction presented in this paper belongs to that latter trend so we will focus on improvements to existing OCRs.

Creating an OCR algorithm is a very complex task. Thus it is sometimes more convenient to improve an existing algorithm. This can also be useful for closed source OCR algorithms or to adapt a generic algorithm to a specific use case. The interested reader can find a thorough review of the state of the art for post-processing techniques in [4].

Some recent pre-processing works include the improvement of training [5] and improving the binarization [6].

Kae et al. [7] run Tesseract on a document to detect a set of reliable characters. Then they use SIFT as a character descriptor and an SVM classifier to OCRize the document. They reduce Tesseract's error rate by 20% on 10 documents.

Most generic approaches use either a lexicon, a confusion matrix or reduce the character space to remove some confusions. An early work is that of [8] which does a thorough study of OCR post-processing issues.

Reynaert [9] reduces the character space of documents and ignores digits. Then he creates a set of word variants (including anagrams) within a given edit distance of a lexicon. Then he uses a custom sorting algorithm to replace OCR errors with the proper word variant. The algorithm called TICCL can detect between 55% and 89% of OCR errors on a corpus of historical Dutch newspapers.

Niklas [4] combines the work of Reynaert with a new word hashing algorithm called OCR-Key. It replaces the anagram extension process by a word similarity process which is more computationally efficient and the hash is based on predefined character classes. Several heuristics are then used to compute OCR corrections. He achieves an error reduction rate between 39% and 75% on a corpus made of several issues of The Times newspaper between 1835 and 1985.

A lot has been done to improve OCR accuracy. Our case deals with printed, modern documents which is considered by everyone as a solved problem with no challenge. However we take a very different standpoint from the state of the art by focusing on the stability and not the accuracy of OCR algorithms. This notion of stability is absent from the state of the art. We will now introduce it.

IV. Formal definition of the notion of stability

One can consider that a stable algorithm is an algorithm capable of producing similar (respectively dissimilar) outputs given similar (respectively dissimilar) inputs. Notice the absence of any ground truth in this definition. We consider that an algorithm is a specific kind of function.

Definition IV.1 (Stable function). *Let us have*
- *A function f (the algorithm): $f : I \longmapsto A$*
- *A binary similarity function s_1 for its input space I and a binary similarity function s_2 for its output space A*

f is stable with respect to s_1 and s_2 if and only if

$$\forall \{a, b\} \in I^2, s_2(f(a), f(b)) = s_1(a, b) \tag{1}$$

In our case, s_1 tests if two documents are similar modulo print and scan noise and s_2 tests if two digests are a match.

Before defining the evaluation metrics, let us summarize what we need to verify the definition of a stable function f:
- A similarity function for the input space: s_1
- A similarity function for the output space: s_2
- A set of similar and different inputs

The functions $s1$ and $s2$ depend on the space in which they are defined but, to the extent possible, they should be made independent of f in order to keep a generic definition of stability.

Now, we can measure how much stable is a function e.g. we want to measure how much Definition IV.1 is true. Since this definition can only take a Boolean value, we will instead measure how frequently it is true. More precisely, given two inputs a and b what is the probability that $s2(f(a), f(b)) = s1(a, b)$? This probability needs to be estimated with a dataset of reasonable size.

Hence, we define a positive and a negative condition. A positive condition occurs when the inputs are similar and a negative condition occurs when they are not. This should not be confused with many medical or security related conventions where a test is said to be positive when the outcome is not equal/not normal.

The similarity of the algorithm's output can be considered as a prediction. It is a true prediction if the output similarity/positiveness is the same as that of the inputs e.g. if Equation (1) holds, and false otherwise. The question becomes: what is the probability that the prediction matches the condition e.g. that it is true? Several classical metrics have already been defined to estimate this probability on a given dataset. Among them we choose a set of four metrics:
- The false negative rate (FNR) is the probability that an event is predicted negative when it is positive e.g. that an original document is wrongly detected as modified.

- The false positive rate (FPR) is the probability that an event is predicted positive when it is negative e.g. that a modified document is wrongly detected as original.
- The false omission rate (FOR) is the probability that an event is positive when it is predicted negative e.g. that a document detected as modified is actually original.
- The false discovery rate (FDR) is the probability that an event is negative when it is predicted positive e.g. that a document detected as original is actually modified.

Other than providing useful probability information, this set of metrics is independent of the dataset bias towards positive or negative conditions. They are computed as:

$$FNR = \frac{\sum \text{False negative}}{\sum \text{Condition positive}} \tag{2}$$

$$FPR = \frac{\sum \text{False positive}}{\sum \text{Condition negative}} \tag{3}$$

$$FOR = \frac{\sum \text{False negative}}{\sum \text{Prediction negative}} \tag{4}$$

$$FDR = \frac{\sum \text{False positive}}{\sum \text{Prediction positive}} \tag{5}$$

They all require the ground truth of similar/dissimilar inputs to be computed during testing. However, once they are computed for a given algorithm, their value (already computed during testing) can be used in the following cases. FOR and FDR provide an estimate of the veracity of the prediction for a given prediction result (without knowing the ground truth) and thus are widely used in commercial applications. FNR and FPR estimate the veracity of the prediction for a given condition (with knowledge of the ground truth) and are thus used to predict the performance of an algorithm on a given dataset. Ideally, they should all be below 5 %. We will now properly define the problem at hand.

V. Alphabet reduction

When reading a password it is frequently difficult to differentiate an O from a 0 or an I from an l. Furthermore, replacing one of these character by the other in a text does not change its readability, but for reference numbers and other codes in which case there will also be an ambiguity for a human reader. Thus it seems pointless to ask an OCR algorithm to differentiate these characters. Furthermore, they introduce a visual ambiguity which will significantly reduce the stability of the output of OCR algorithms.

The alphabet reduction [2] is a post-processing that removes the ambiguities contained in OCR algorithms. This leads to a problem that is better posed and is easier to solve. Contrarily to most works who focus on the algorithm and the text, it focuses on the observer and what is meaningful for him. Hence this approach is unsupervised and based on human vision.

Table I shows the character classes that are projected onto the same character. Because it is difficult to identify the number of empty lines and sometimes the spacing between lines, they are removed. Once again this does not change the readability of the text. Similarly, differentiating between a tabulation and a certain number of spaces is difficult, thus they are removed. A similar principle is applied for the other characters that are visually difficult to distinguish. The choice of these projections was based on Tesseract's default English alphabet and on the main OCR errors that could have also been made by a human.

Character	Replacement
Empty line	Removed
Tabulation and space	Removed
— (long hyphen)	- (short hyphen)
', ‘ (left and right apostrophes)	’ (centered apostrophe)
", “, ” (left and right quotes, double apostrophe)	” (centered quote)
I, l, 1 (capital i, 12^{th} letter of the alphabet, number 1)	|(vertical bar)
O (capital o)	0 (zero)
fi (ligature)	fi (two letters f and i)
fl (ligature)	fl (two letters f and l)

TABLE I: Alphabet reduction

This alphabet reduction could lead to projecting two different words onto the same word, which could introduce a possibility for undetected modifications of the textual content of a document. In [2] the only reported collisions where those due the confusion between an I (a capital i) and an l (a lower case L). In such cases, the confusion is possible but would not make a meaningful sentence. When differentiating upper and lower case letters, the collision probability is 0.0002. This guarantees that the level of security of the system is preserved.

VI. Evaluation of the alphabet reduction

We compare the stability of two state of the art OCRs: Tesseract and Finereader Engine 11 without and with alphabet reduction. For both algorithms we use the initial English training provided with them. For this we will first present the test dataset, then an additional metric and finally the evaluation results.

A. Testing dataset: L3iTextCopies

We use the same dataset as in [2]. It contains clean, text-only, printed documents with a single or double column layout and with only and all the characters that they both can recognize (the limitation comes from Tesseract).

The dataset is made of 22 pages of text with the following characteristics:

- 1 page of a scientific article with a single column header and a double column body
- 3 pages of scientific articles with a double column layout
- 2 pages of programming code with a single column layout
- 4 pages of a novel with a single column layout
- 2 pages of legal texts with a single column layout
- 4 pages of invoices with a single column layout
- 4 pages of payslips with a single column layout
- 2 pages of birth extract with a single column layout

It has several variants of these 22 text pages by combining:

Scanner	150dpi	300dpi	600dpi
Konica Minolta Bizhub 223		X	XX
Fujitsu fi-6800	XXX	X	
Konica Minolta Bizhub C364e		X	X

TABLE II: Scanning resolution for each scanner, one "X" per scan

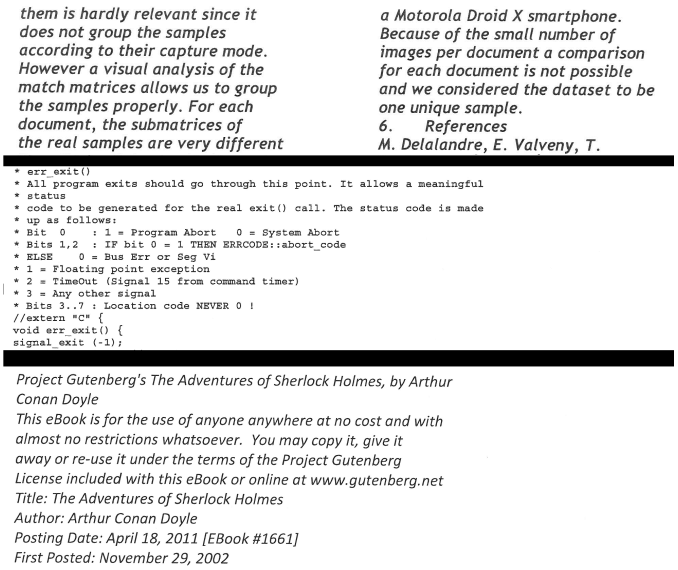

Fig. 2: Three excerpts of document images of the dataset

- 6 fonts : Arial, Calibri, Courier, Times New Roman, Trebuchet and Verdana
- 3 font sizes : 8, 10 and 12 points
- 4 emphases : normal, bold, italic and the combination of bold and italic

This makes 1584 documents. They were printed with three printers (a Konica Minolta Bizhub 223, a Sharp MX M904 and a Sharp MX M850) and scanned with three scanners and at different resolutions between 150 dpi and 600 dpi as shown in Table II. This makes a dataset of 42768 document images. Figure 2 shows an example of the images contained in the dataset.

B. Metrics

In order to use the stability metrics (FNR, FPR, FOR and FDR) we need to define the input and output similarity functions. The input similarity function, s_1, is the indicator of whether the images are copies of the same document at the same resolution. Thus to be considered as identical, two images will need to have the same text, font, font size, font emphasis and resolution. Such a precise criterion will allow us to study the impact of each parameter (font, font size, font emphasis and resolution). The output similarity function, s_2, is the binary comparison of the OCR outputs. The FNR, FPR, FOR and FDR should all be as low as possible to have a stable algorithm.

For reference purposes, we also use a common metric for the evaluation of OCR performance: the character error rate. It is defined by:

$$err = 1 - \frac{nc_{cr}}{nc_t} \tag{6}$$

where nc_{cr} is the number of correctly recognized characters and nc_t is the total number of characters. It is computed at a character level contrarily to the other metrics that are computed at the page level.

To evaluate the OCR error rate we need a ground truth for the original text. In the case where we use the alphabet reduction as a post-processing step for the OCR output, we will also apply it on the ground truth so that the evaluation is consistent.

C. Results

Table III shows the results for both OCR algorithms without and with the alphabet reduction post-processing. The text variations are such that there are no false positives and the false positive and discovery rates (FPR and FDR) are both always equal to 0. Because of the strong balance of the dataset towards negative conditions, the discriminative criterion is the FNR. The FOR is always below 0.02%.

Clearly, Finereader is better than Tesseract at all resolutions. Tesseract has definitely more difficulties dealing with images at 150 dpi as its error rate raises up above 8% while it remains below 2% at the other resolutions. Finereader on the other hand has a very slightly higher error rate at 600 dpi than at 300 dpi. Both facts are likely related to the default algorithm training being performed on images at 300 dpi for both algorithms. In the case of Tesseract, this may also come from a technical limitation. Tesseract uses a contour approximation algorithm to recognize characters. It is possible that at 150 dpi this algorithm is not able to approximate the contours appropriately because of the low resolution.

The alphabet reduction reduces the FNR by approximately 20 points, e.g. from 84% to 65% for Tesseract at 600 dpi. The improvement is lower at 150 dpi in particular for Tesseract. This is because the error rate is higher at this resolution and thus there are more sources of instability that are not related to recognition ambiguity. The alphabet reduction also reduces the character error rate by 50% except for Tesseract at 150 dpi.

We can also notice that while increasing the resolution from 300 to 600 dpi does not change the error rate much, it reduces the FNR for both algorithms. This shows that both algorithms do not make less mistakes, but make mistakes in a more repetitive manner.

A more detailed analysis of the errors shows that:

- The font size (in pts) has a similar effect as the resolution.
- The font size (from the font design) can also reduce the performance, in particular Times New Roman and Courier.
- Italic poses problems. This is due to the ambiguity between "/" and the italic "I" (capital i).
- The pages of code are badly recognized because they have more out of dictionary words and an unusual syntax.

		Original algorithm			With alphabet reduction			
Metrics	Algorithms	150 dpi	300 dpi	600 dpi	150 dpi	300 dpi	600 dpi	BCS
FNR (%)	Tesseract	89	87	85	88	70	65	50
	Finereader	**76**	**70**	**68**	**62**	**50**	**48**	**31**
FOR (%)	Tesseract	0.019	0.018	0.018	0.019	0.015	0.014	0.010
	Finereader	**0.016**	**0.015**	**0.014**	**0.013**	**0.011**	**0.010**	**0.007**
char err (%)	Tesseract	8.7	1.6	1.5	8.1	0.7	0.6	**0.2**
	Finereader	**1.5**	**1.1**	**1.2**	**0.8**	**0.5**	**0.6**	0.3

TABLE III: Performance of the OCR algorithms and of the alphabet reduction. "BCS" stands for "best case scenario". The figures in bold are the best results between the two algorithms.

	Tesseract		Finereader	
Stability with respect to	All cases	BCS	All cases	BCS
Printer and scanner	74	50	53	31
+Resolution	90	72	62	34
+Emphasis	96	77	69	40
+Font	98	83	75	46
+Font size	98	84	76	46

TABLE IV: Influence of input similarity criterion on the FNR performance of the OCR algorithms with alphabet reduction. Values are in percentage. "BCS" stands for "best case scenario".

- The dotted lines in forms are badly recognized for the same reason.

Thus we devise a more reasonable best case scenario without italic emphasis, with a resolution of at least 300 dpi, with a font size of at least 10 points, with all fonts but Times New Roman and Courier and with no pages of code. In this case we reach an FNR of 30% with Finereader. While this is far from the goal of being below 5%, this is also far better from the initial 75-90%.

Since the OCR algorithms focus on the text content without taking into account the font type, size or emphasis or the image resolution it should be possible to have an input similarity function that does not take these into account either. In order to study the influence of taking into account each parameter we have incrementally relaxed the input similarity function. Table IV shows the FNR variation with respect to the similarity function and criterion for both algorithms with the alphabet reduction.

As we can see, the more we relax the similarity criterion, the more unstable the algorithms are. This is because there are more errors for each set of images of a same document. Thus there are more combinations and more instability possibilities. However, it seems that relaxing the font size does not increase the instability by more than 1%. This is probably linked to the low error rate, but nevertheless it shows that evaluating the stability of an OCR on one font size could be sufficient provided that the other parameters are varied enough. Considering our best case scenario and usual character sizes, a size of 10 points could be a good choice.

We can also notice that the FNR varies more in the general case than in the best case scenario and Tesseract's FNR degrades twice more easily than the one of Finereader: it increases by 34 points in the best case scenario while the increase is only of 15 points for Finereader. This contributes to showing that Finereader has a superior stability.

VII. Conclusion

In this paper we have formalized the definition of a stable algorithm and how to evaluate it. We used this to study the stability of two OCR algorithms with respect to several parameters: the font, the font size, the font emphasis, and the image resolution.

Producing a stable algorithm requires removing as many ambiguities as possible. This is the goal of the alphabet reduction, which is based on a study of the observer and of human character recognition. Thus, its performance is not influenced by the content being processed. With it, visually similar characters are considered as the same character. This simple post-processing halves the character error rate and reduces the FNR by 20 points. To our knowledge no other post-processing achieves these results while being content agnostic. This post-processing is currently limited to Latin characters, but we expect that it can be extended in the future.

We have benchmarked two state of the art algorithms: Tesseract and Finereader. Both are limited by the FNR performance. Finereader clearly stands out as the most accurate, versatile and stable algorithm. If the only image variation allowed is the printer and scanner hardware, it reaches an FNR of 31%. This goes up to 46% when every variation is allowed. Hence it is still very unstable.

The performances presented here are obtained on a fairly clean dataset as shown by the very low error rate (below 1%). Thus the algorithms may be less stable on noisier images.

The last consequence of this study is that the issue of having a high quality stable OCR for printed English text is far from being solved, despite the common belief that this issue does not present a challenge anymore. The current best way to extract text in a stable manner is to use FineReader and apply the proposed alphabet reduction on it. It works best if the text is not in italic and for character sizes above the one of the Arial font at 10 points. Text should also be scanned at a resolution of at least 300 dpi. Also text with unusual syntax such as programming code will yield worse results.

ACKNOWLEDGMENT

This work is financed by the ANR (French national research agency) project SHADES referenced under ANR-14-CE28-0022.

REFERENCES

[1] A. Smith, "Identity fraud: a study," Economic and Domestic Secretariat Cabinet Office, Tech. Rep. July, 2002.

[2] S. Eskenazi, P. Gomez-Krämer, and J.-M. Ogier, "When document security brings new challenges to document analysis," in *International Workshop on Computational Forensics*. SPIE, 2015, pp. 104–116.

[3] J. A. Sanchez, A. H. Toselli, V. Romero *et al.*, "ICDAR 2015 Competition HTRtS : Handwritten Text Recognition on the tranScriptorium dataset," in *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 1166–1170.

[4] K. Niklas, "Unsupervised post-correction of OCR Errors," Ph.D. dissertation, Leibniz Universität Hannover, 2010.

[5] A. Agarwal, R. Garg, and S. Chaudhury, "Greedy search for active learning of OCR," in *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2013, pp. 837–841.

[6] T. Chattopadhyay, V. R. Reddy, and U. Garain, "Automatic selection of binarization method for robust OCR," in *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, aug 2013, pp. 1170–1174.

[7] A. Kae, G. Huang, C. Doersch *et al.*, "Improving state-of-the-art OCR through high-precision document specific modeling," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Comput. Soc. Press, 2010, pp. 1935–1942.

[8] W. S. Rosenbaum and J. J. Hilliard, "Multifont OCR postprocessing system," *IBM Journal of Research and Development*, vol. 19, no. 4, pp. 398–421, jul 1975.

[9] M. Reynaert, "Non-interactive OCR post-correction for giga-scale digitization projects," *Computational Linguistics and Intelligent Text Processing*, vol. 4919, pp. 617–630, 2008.