Combination of Optical Character Recognition Engines for Documents Containing Sparse Text and Alphanumeric Codes

Iago Lourenço Correa, Paulo Lilles Jorge Drews-Jr and Ricardo Nagel Rodrigues
Center for Computational Science - C3
Federal University of Rio Grande (FURG)
Rio Grande - RS, Brazil
Email: iago.correa@outlook.com, paulodrews@furg.br, ricardonagel@gmail.com

***Abstract*—Many companies that buy machines, parts, or tools retain documents such as notes, receipts, forms, or instruction manuals over the years, and they may find themselves in need of digitizing these accumulated documents. Thus, when using optical character recognition (OCR) systems in these documents, it is possible to note that these systems can present two main difficulties. The first is to locate the sparse text in a non-continuous way, and the second is to match words that are closer to codes and less to words in human language. Although there are many works in the literature about sparse texts, such as forms and tables, there is usually not much concern about the issue with codes in which one can not rely on dictionaries or even both problems together. Therefore, to correct this issue without having to search for extensive databases or conduct training and development of new models, this work proposed to take advantage of pre-trained models of OCR such as from the Tesseract engine or the Google Cloud's Vision API. In order to do so, we proposed the exploration of combination strategies, including a new one based on median string. The experimental results achieved up to 3.09% improvement in character accuracy and 1.16% in word accuracy in comparison to the best individual performances from the engines when our method based on string combination was adopted.**

I. Introduction

Optical character recognition (OCR) systems can transform text in image format to data that can be processed by a machine. One of the first OCR systems was developed in the 40s [1], but there are still open problems in the character recognition field. For example, the recognition of text from visual structures such as tables and boxes [2], or the recognition of handwritten characters in different daily scenarios [1]. But not only that, other problems can be found.

Many companies that buy machines, parts, or tools retain documents such as notes, receipts, forms, or instruction manuals over the years, and they may find themselves in need of digitizing these accumulated documents. But when using OCR systems for recognizing these documents, it is possible to note that they present two main difficulties. The first is to locate the sparse text in a non-continuous way, and the second is to recognize words that are closer to codes and less to words of the human language.

In documents with sparse text, grammar rules can not be applied and it is common to find visual structures that usually can confuse the OCR, such as tables, graphics, boxes, or headers. Besides that, alphanumeric identification codes can also be found. That kind of code does not configure a word that would normally be in a post-processing method based on a dictionary. It is a code composed of a combination of alphabetical characters and numbers, such as "SP01" or "EZ10HD05".

In this context, this work proposes to take advantage of several pre-trained models of OCR such as from the Tesseract engine or the Google Cloud's Vision API, for example, in documents that resemble the ones discussed before, with sparse text and alphanumeric codes. In order to do so, this work presents a new method based on combination strategies, along with their validation and comparison to individual results. The strategies explored are the majority voting, ranking, and the median string, which was proposed by Kohonen [3] and is newly proposed in the context of OCR engines combination in this work.

The work is structured in four sections. First, Section II presents the main references for this work. Then, Section III presents the proposed methodology. Section IV presents the results for the different experiments conducted. And finally, there are the conclusions of this work.

II. Related Work

In reason of the development of knowledge in the fields of pattern recognition many different approaches have been explored in OCR systems. Currently, the state of the art is given by the adoption of deep neural networks, but it is possible to note that different models can stand out in different contexts [1]. Even so, some works adopting combination strategies can be found. Those combinations are usually presented in two different levels: feature level, and decision level [4]. In the feature level combination, a feature extractor model is adopted before the final classification into a character. And in the decision level, the combination happens in the class selection level, i.e. when all the systems to be combined are already decided for one class each. In this work, decision level combinations are adopted.

In the domain of recognizing characters from a video, Petrova et al. [5] present a weighted combination of the

recognition from different frames from a video. The weights adopted are a focus estimation of the image and an a-posteriori recognition confidence, and with this methodology, the results indicate improvement in the recognition of text in videos.

In the recognition of handwritten Devanagari numerals, Singh et al. [6] propose a feature extraction approach based on Information theory measures, followed by the decision level combination between different multilayer perceptrons. For the decision level combination, it was adopted three different approaches: majority voting, decision templates, and Dempster Shafer [7], in which, the last one achieved the best results. The Dempster Shafer combination approach was not adopted in this work because it requires the probability for each possible class, information that is not available for all OCR engines.

Looking for a way to combine the capabilities from different OCR systems, Boiangiu et al. [8] propose the combination of different OCR outputs through a voting system. In their work, the same image is pre-processed with different techniques generating different pre-processed images which are then processed in the OCR system. The outputs for those different techniques are combined. Their proposed methodology achieved improvement between 4% and 5% of accuracy in the recognized text.

Petrescu et al. [9] present the combination for the output of two OCR engines, the Tesseract and the Asprise. For this combination, it is adopted two different weights in voting. The first one is the confidence given by the engines and the second one is obtained through an accuracy calculated in a document that is similar to the one that is being processed, i. e. it is needed to have annotated documents. The experiment conducted by them, indicated improvement in comparison to the isolated classifiers. Similarly, Dannélis et al. [10] also present the combination of the output for two OCR engines, the Tesseract and the ABBYY Finereader, to improve the text identification in historical documents and newspapers in a Swedish library. In their work, the combination is given by selecting the words in the outputs with greater confidence.

Searching for improvement in text from the 15^{th} and 16^{th} centuries, Reul et al. [11] propose the combination of different models from the OCRopus engine. In their work, a dataset is divided into subsets that are used to retrain the model from OCRopus, the outputs obtained from those models are combined through voting weighted by the models confidence. With this methodology, it was obtained a reduction of 50% of errors in the text identification.

As noted by Zeni and Jung [12] in the problem of automatic license plate recognition, no grammatical or semantic information relates the characters from the plate, so it is a different problem from the generic OCR problem. In their work, they proposed a weakly supervised model that achieved competitive results in character detection. In the same problem for license plates, Montazzolli and Jung [13] proposed an end-to-end deep learning that correctly recognized all the characters of a license plate in 63.18% of the test set. Also, concerning alphanumeric codes with no semantic or grammatical information, Han et al. [14] propose the adoption of a CNN model for character identification in cables surface achieving average recognition rate of 92.6% when number '0' is considered equal to the letter 'O' and '1' is equal to the letter 'I' .

Finally, although there are some works concerning alphanumeric codes and many works that focus on the sparse text issue [2] [15], there are few works that tackle both problems. Hence, adopting combination strategies to correct these problems can be beneficial, especially given its ease of implementation, without training any new model. Also, the median string concept is newly proposed to combine different OCR engine outputs in this work.

III. Methodology

This work proposes the combination of OCR engines to enhance the individual performance of those. Thus, an improvement in the recognition of characters in documents with sparse text and alphanumeric codes is expected. In this Section, OCR engines, our method based on combination strategies, and finally, the validation data are presented.

A. OCR Engines

This work investigates some OCR engines freely available under the Apache License 2.0. Besides these engines, some results were also generated adopting the Google Cloud Vision OCR[1].

The Tesseract[2] [16] was one of the first OCR engines with open-source code since 2005 and it is one of the oldest OCR engines that is still under development [17]. Therefore, it is a mature system that is vastly known and was adopted in many different works, such as in [9] and [10]. Its most recent stable version $4.1.1$ was launched on December 26^{th}, 2019, and it works based on a long short-term memory network (LSTM) since its version $4.0.0$ launched in 2016. In this work, the results were also analyzed for other two different Tesseract versions: 3.05 (legacy version, it does not adopt LSTM models) and $5.0.0$ (alpha version). Since this work searches for the combination of OCR systems, it is expected that better results could be achieved with the adoption of the the different Tesseract versions in the same combination.

The OCRopus[3] engine initially was a set of tools for OCR based on the Tesseract, however, since 2010 the OCRopus OCR adopted a LSTM model [17]. Currently, its implementation consists of a collection of software to analyze documents, besides binarization, layout analysis, error measurement, and confusion matrix calculations functions for example. The kraken[4] engine started in 2015 as a fork from OCRopus and currently its implementation have changed enough to not be considered only a ramification. Its character recognition is based on a RNN model. The Calamari[5] engine has been developed since 2018, based on the OCRopus and kraken, but backed up by TensorFlow[6] [17]. This engine was projected

[1]https://cloud.google.com/vision/docs/ocr?hl=pt-br
[2]https://github.com/tesseract-ocr/
[3]https://github.com/tmbarchive/ocropy
[4]http://kraken.re/
[5]https://github.com/Calamari-OCR/calamari
[6]https://www.tensorflow.org/

to work through command lines, and the system adopts the segmentation codes from OCRopus.

Finally, the Google company has a line of products and services towards machine learning, which includes a computational vision API in the cloud, the Google Cloud's Vision API. Also, included in the API services, there is an OCR system. Different from the other engines of this work, it is not possible to train a model with your data. Besides that, it is not an open-source project, its free usage is limited to 1000 images per month. Hence, it is interesting to analyze its capacity in comparison to open-source OCR engines. Due to its usage limitation, in this work, the Google Vision OCR was just applied in one of two adopted datasets.

B. Combination-based method

According to Mohandes et al. [4], the combination of classifiers can be achieved at different levels, with no single strategy being the best. Thus, different strategies of combination between the OCR systems have been presented and evaluated.

The first combination strategy follows the idea from Petrescu et al. [9], in which confidence values from the engines weight the different outputs, ranking them. Following the work from Singh et al. [6] and Xu et al. [18], the majority voting was explored. And in the end, it was also explored the median string calculation proposed by Kohonen [3], which has not been explored before for the purpose of combining OCR engines.

Although there are many works in the literature about the combination in different levels, many of them recently have been adopting feature level combination i.e. with one classifier as a feature extractor and the other for the character recognition. This approach does not fit for the combination of closed systems such as OCR engines, where there are one input image and the final output, being only possible the combination at the decision level [4].

The proposed method is divided in three steps:

1) Segmentation: The images from the chosen datasets are complete documents, containing many lines of text not continuously. Therefore, if each OCR engine segment the images, different text regions may be found, making it harder to compare those regions due to a possible lack of equivalency. For that reason, the combination experiments between the OCR systems were performed utilizing line-level segmented images as input. Yet, the engines still perform segmentation at word and character levels.

Therefore, considering the fact that the Tesseract has a parameter that changes the segmentation mode, the *psm*, in the experiments, it was evaluated the possibility to combine different outputs from the Tesseract for different segmentation modes. The possible values for the *psm* parameter are in Table I with their respective meaning. In this work, the values 0 and 2 are not utilized, because those modes do not perform character recognition.

2) Character Alignment: For a same image, different engines may have different size outputs because the engines segment the characters differently. Therefore, in one image containing one character, it can be wrongly recognized as two characters or even none, for example, where it should be identified as an "m" it can be identified as "rn". So, to combine the outputs from different OCR systems, all outputs must be aligned at a character level to identify and match those gaps generated by mistakenly identified characters.

Therefore, as a way to match those gaps generated by segmentation in different engines, we adopted a sequence alignment strategy. Through the alignment, identical characters tend to be aligned and the difference between the outputs from the OCR systems can be managed accordingly. As presented before, considering that the dataset is composed of full documents with many text lines, not only characters, it is expected the alignment to be beneficial.

The alignment task follows the method proposed by Katoh et al. [19] that aligns multiple character sequences based on fast Fourier transform[7]. For the adoption of this implementation, spaces between words were treated as a special character, because the input for the engines were text lines, not words. In Figure 1 one text line segmented from SROIE dataset is illustrated. Table II presents the outputs for three different OCR engines for that image as it is and aligned, where the symbol Ø denotes one gap where no character is expected. It can be observed, that more characters are matched after the alignment. Also, the adoption of the alignment makes it easier to combine different size strings, because it defines where gaps can be located and which character is compared to which.

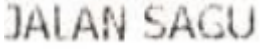


Fig. 1. Example of a segment from a document of the SROIE dataset.

3) Combinations: Regarding the last step, three different combination approaches were explored.

a) Majority Voting: It is one of the most utilized non-trainable combination strategies [6]. In the majority voting, it is counted the number of classifiers that vote for a particular class, and the one with the majority of the votes is selected. In this work, the classes are the identified characters, after the alignment, for each position of the strings obtained by the engines, it is voted and chosen the most recurrent character. Also, in case of a tie, it is picked any of the characters.

b) Ranking: In this combination approach, it is utilized one value to weight the classes obtained by the classifiers. By the sum of those values, the classes are ranked, and the one in the first place is chosen. In this work, the classes are the characters, and the weight values are the confidence obtained from the OCR engines. So, for each position of the aligned strings output by the engines, a rank is done according to the confidence. The character in the highest position is chosen.

Table III presents one example of this combination approach for three strings. In the table, the confidence for each character is presented, as well as for the gaps created in the aligned process, the confidence of 0.50 was adopted as default. With

[7]https://mafft.cbrc.jp/alignment/software/

TABLE I
PAGE SEGMENTATION MODES FROM THE TESSERACT, DEFINED THE THE PARAMETER *psm*.

psm	**Description**
0	Orientation and script detection (OSD) only
1	Automatic page segmentation with OSD
2	Automatic page segmentation, but no OSD, or OCR
3	Fully automatic page segmenation, but no OSD (default value)
4	Assume a single column of text of variable sizes
5	Assume a single uniform block of vertically aligned text
6	Assume a single uniform block of text.
7	Treat the image as a single text line
8	Treat the image as a single word
9	Treat the image as a single word in a circle
10	Treat the image as a single character
11	Sparse text, find as much text as possible in no particular order
12	Sparse text with OSD
13	Raw line, treat the image as a single line, bypasssing hacks that are Tesseract-specific

TABLE II
ALIGNED OUTPUT FROM THREE DIFFERENT OCR ENGINES.

OCR Engine	Identified characters										
Calamari	A	A		L	A	C	U	Ø	Ø		
kraken	I	A	A	N		A	L	U	Ø		
OCRopus	W	A	N		S	-	A	L	U		
	Aligned characters										
Calamari	Ø	Ø	A	A		Ø	Ø	Ø	L	A	C	U
kraken	I	A	A	N		Ø	Ø	Ø	Ø	A	L	U
OCRopus	Ø	W	A	N		S	-	A	L	U	Ø	Ø

TABLE III
RANKING COMBINATION EXAMPLE.

String 1	Character	Ø	m	a	t	t	a
	Confidence	0.50	0.90	0.85	0.50	0.30	0.80
String 2	Character	r	n	a	t	2	Ø
	Confidence	0.50	0.40	0.80	0.75	0.45	0.50
String 3	Character	Ø	m	2	t	a	Ø
	Confidence	0.50	0.80	0.40	0.70	0.85	0.50
Rank		Ø, 1.00 r, 0.50	m, 1.70 n, 0.40	a, 1.65 2, 0.4	t, 1.95	a, 0.85 2, 0.45 t, 0.30	Ø, 1 r, 0.80
Final output		Ø	m	a	t	a	Ø

the characters and the confidence, a rank is made for each position, and then it is chosen the character with the highest sum of confidence, i. e. the highest position in the rank.

c) Median String: Originally proposed by Kohonen [3] with the objective to smooth or average over repeated erroneous strings, the median string is explored as an approach to combine OCR engines in this work. This median string can be from a set or generalized. The first one is defined by the element in the set of strings which the sum of distances is the smallest between the elements, and the second one is a hypothetical element created to minimize even more that distance.

Therefore, the set median can be found by calculating all the distances between elements and picking the minimum sum of distances. On the other hand, the generalized median is found through the systematic variation of each character in the set median, creating errors through the whole alphabet and searching for a minimal sum of distances. In this work, it is adopted the generalized concept. And also, different elements can present different weights to ponder the distances. Later in this work, the median string that utilizes the OCR confidences as weights is referenced as "wei_median", while the median string without weights is called only "median".

To calculate the median it is also necessary to adopt a distance measurement. In this work, we adopted the Levenshtein distance, which according to Kohonen [3] is capable of yielding better results. In simple terms, the Levenshtein distance can be understood as the minimum number of basic operations to transform one string into another. These basic operations are the insertion, deletion, and substitution of one character.

C. Validation Data

For the validation of the proposed methodology, evaluation metrics were calculated with two different datasets. The first one is the Form Understanding in Noisy Scanned Documents[8] [20] (FUNSD), which is a dataset composed of 199 forms completely annotated. This dataset is originally intended for evaluating form understanding.

The second one is the Scanned Receipts OCR and Information Extraction [9] [21] (SROIE) dataset, which was presented at a competition in ICDAR 2019 focusing on information extraction on scanned receipts and containing 986 documents. Although the methodologies explored in the competition were trying to achieve new models and processing strategies, in this work, we explored the potential of improving pre-trained models through combination.

Although the datasets present different document types, forms and receipts, they were chosen because they present sparse text and alphanumeric codes. Besides that, both contain annotated ground-truth where it can be found the correct text output for each document and its location.

IV. RESULTS

According to the methodology presented previously, this section explains the evaluation metrics, presents the results obtained through different experiments, and discuss those results. The first experiment presents the analysis of the individual performance of the OCR engines followed by the results for the combination strategies.

[8]https://guillaumejaume.github.io/FUNSD/

[9]https://rrc.cvc.uab.es/?ch=13

A. Validation Metrics

Following other works that analyze the performance of OCR systems, the metrics evaluated in this work are character accuracy, word accuracy, and the number of errors in terms of edit operations. The metrics were calculated through the ocreval[10] toolset, which is a updated implementation of a toolset to assess OCR systems called ISRI [22]. This toolset was also adopted in other works, such as the work of Reul et al. [11].

1) Character Accuracy: The recognized text is compared to the correct text to determine the minimal number of operations necessary to correct the generated text, this quantity is the number of errors e. If there are n characters in the correct text, the accuracy per character is given in the Equation 1 according to Rice et al. [23].

$$\text{character accuracy} = \frac{n - e}{n}. \tag{1}$$

2) Word Accuracy: Through comparison of the recognized text and the correct text, the word accuracy is defined by the percentage of words correctly identified [23]. For a word to be correct, it must contain all the characters of the word in the correct text, but uppercase and lowercase letters are not discriminated.

3) Edit Operations: The edit operations are the insertion, deletion, and substitution of one character to transform one string into another, i.e. the recognized text into the correct one. Through this metric, it is expected to understand if unnecessary information is being acquired by the number of deletions, or even, if the information is being lost by the number of insertions, for example.

B. Individual Performance

First, it is necessary to analyze the individual performance of the OCR engines in order to compare with the results from the combinations later. In this experiment, no training was performed and all engines use standard models for the English language. The obtained results for all the adopted engines can be observed in Table IV for the FUNSD dataset and Table V for the SROIE. In the case of Tesseract 4.1 and 5.0, results were generated for all the possibles values of *psm*, but only the best results for each version are presented in both tables.

In terms of individual performance in the FUNSD dataset, the best accuracy results were achieved with the Tesseract 5.0 $psm = 6$ (assume a single uniform block of text) for the character accuracy, and with the Tesseract 4.1 $psm = 6$ for the word accuracy. The best results in terms of edit operations were also achieved with those Tesseracts. It is interesting to note that the Tesseract achieved results greater than all the other engines.

For the SROIE dataset, the best results are achieved with the Google Vision OCR, in terms of the accuracies and most of the edit operations, except for the number of insertions. The Tesseract 5.0 $psm = 13$ (treat the image as a raw line) achieved a smaller number of insertions than the Google Vision. Here it is important to note that the usage of the Google Vision API is not completely free, being limited to a certain number of images per month, and even so, the Tesseract achieved a character accuracy very close to the Google Vision one.

[10]https://github.com/eddieantonio/ocreval

TABLE IV
OCR ENGINES INDIVIDUAL PERFORMANCE ON FUNSD.

	FUNSD				
	Character accuracy	Word accuracy	Edit operations		
			Insertion	Deletion	Substitution
Calamari	43.56	23.46	83461	10892	23368
kraken	45.18	13.67	83052	8757	27038
OCRopus	36.56	16.74	91073	17600	29695
Tesseract 5.0 psm = 6	**86.82**	80.62	12920	**7515**	**6135**
Tesseract 4.1 psm = 6	86.54	**81.11**	**12406**	8710	6142
Tesseract 3 psm = 12	34.82	17.14	99169	27708	27834

TABLE V
OCR ENGINES INDIVIDUAL PERFORMANCE ON SROIE.

	SROIE				
	Character accuracy	Word accuracy	Edit operations		
			Insertion	Deletion	Substitution
Calamari	64.55	56.31	197891	171892	42461
kraken	53.16	36.56	253882	177690	71147
OCRopus	61.21	48.34	207288	184269	59538
Tesseract 5.0 psm = 13	71.18	79.73	**166371**	166117	24168
Tesseract 4.1 psm = 10	70.57	78.42	168968	171787	25162
Tesseract 3 psm = 12	15.40	11.49	533505	563411	34199
Google Vision	**71.25**	**86.08**	175953	**150593**	**14947**

C. Combination Performance

Our combination strategies were investigated in two different moments in this work. The first one with the Tesseract parameter fixed $psm = 12$ (sparse text with OSD) because it was expected to yield better results in documents with sparse text. And the second one happened using different values of psm (1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13 as observed from Table I) as different OCR systems in the combinations and also using assumptions based on the first results generated with fixed psm value.

In the following tables, we adopted a simplified nomenclature for each combination for better readability. This nomenclature starts by an abbreviation of the combination type ("maj", "rank", "median" or "wei_median") and it is followed abbreviations of the names from the OCR systems. For the Tesseract this abbreviation is like "t4p12", where the number following the letter 't' is the version of the Tesseract and the number following 'p' is the psm parameter value. Also, "tess_all" stands for all Tesseract versions with all the possible psm values, and "tess_4" or "tess_5" stands for all possible psm values for a specific version of the Terreseract, $4.1.1$ and $5.0.0$ respectively.

1) Results without psm variation: In this experiment majority voting, ranking, median string, and median string with

TABLE VI
COMBINATION RESULTS ON FUNSD WITHOUT *psm* VARIATION (*psm* = 12).

	FUNSD				
	Character accuracy	Word accuracy	Edit operations		
			Insertion	Deletion	Substitution
Best individual (*psm* = 12)	82.51	74.42	20488	5214	6766
maj_t3p12_t4p12_t5p12	82.49	74.12	20974	4606	6743
maj_t3p12_t4p12_t5p12 _kra_cal_ocro	71.10	49.43	42506	5284	11977
rank_t3p12_t4p12_t5p12	65.75	40.59	35020	28736	20767
rank_t3p12_t4p12_t5p12 _kra_cal_ocro	49.67	21.44	35228	57053	40572
median_t3p12_t4p12_t5p12	81.51	72.68	24001	**3814**	**6534**
median_t3p12_t4p12_t5p12 _kra_cal_ocro	67.73	51.12	48954	4251	11042
wei_median_t3p12_t4p12_t5p12	**82.80**	**75.15**	**19720**	5013	6674
wei_median_t3p12_t4p12_t5p12 _kra_cal_ocro	80.12	68.85	25582	6029	8234

TABLE VII
COMBINATION RESULTS ON SROIE WITHOUT *psm* VARIATION (*psm* = 12).

	SROIE				
	Character accuracy	Word accuracy	Edit operations		
			Insertion	Deletion	Substitution
Best individual (*psm* = 12)	**71.25**	**86.08**	175953	**150593**	**14947**
maj_t3p12_t4p12_t5p12	60.68	66.64	223267	170137	35494
maj_t3p12_t4p12_t5p12 _kra_cal_ocro	67.36	68.30	193801	155365	29000
rank_t3p12_t4p12_t5p12	55.88	55.71	227484	186953	53899
rank_t3p12_t4p12_t5p12 _kra_cal_ocro	53.20	43.53	185597	240594	83292
median_t3p12_t4p12_t5p12	59.89	63.99	236195	152272	34025
median_t3p12_t4p12_t5p12 _kra_cal_ocro	65.31	66.41	203327	152807	33266
wei_median_t3p12_t4p12_t5p12	60.92	67.80	218636	170173	34996
wei_median_t3p12_t4p12_t5p12 _kra_cal_ocro	68.16	72.94	185813	159979	28170
maj_t3p12_t4p12_t5p12_kra _cal_ocro_google	70.05	76.80	182885	151969	20257
rank_t3p12_t4p12_t5p12_kra _cal_ocro_google	53.55	43.95	181981	241767	81714
median_t3p12_t4p12_t5p12_kra _cal_ocro_google	69.20	76.22	185218	152539	23045
wei_median_t3p12_t4p12_t5p12 _kra_cal_ocro_google	71.11	81.24	**173962**	156568	19317

confidence weights were explored. The results from this experiment can be fully observed in Table VI for the FUNSD and Table VII for the SROIE.

On the results over FUNSD, it can be observed that our median string combination with confidence weight for three Tesseract versions achieved the best results. Not only this result was the best between the combinations, but it was also better than the best individual result between the combined engines. Therefore, it is verified that there was information gain through the combination. Besides that, the Tesseract 3 individual results were not as good as the other Tesseract versions, so it is expected that other OCR systems in its place could yield even better results.

Over the SROIE dataset, no combination was able to achieve results better than the individual result from Google Vision, not even combinations that used it. However, it is possible to observe that the combinations of all OCR systems except the Google Vision (maj_t3p12_t4p12_t5p12_kra_cal_ocro, median_t3p12_ t4p12_t5p12_kra_cal_ocro and wei_median_t3p12_t4p12_t5 p12_kra_cal_ocro) were capable to improve those individual OCR results. So, there is a capability to achieve some gain, even though those combinations could not improve on the Google Vision results.

TABLE VIII
BEST RESULTS COMPARISON ON FUNSD.

	FUNSD			
	Individual		Combination	
	Tesseract 5.0 psm = 6	Tesseract 4.1 psm = 6	maj_tess_5	wei_median_t4p6 _t5p6_t5p12
Character accuracy	86.82	86.54	**87.82**	87.56
Word accuracy	80.62	81.11	80.03	**81.55**
Insertion	12920	12406	13824	**12069**
Deletion	7515	8710	**5562**	6938
Substitution	6135	6142	**5078**	5695

2) Results with psm variation: In this second experiment, the ranking strategy was not explored, because it did not generate good results in the previous experiment. Besides that, the previous results on combinations and the results from different values of psm oriented the selection of the combination groups, OCR engines which yielded better results were selected to be combined.

The results for character and word accuracy are presented in Figure 2 for the FUNSD and Figure 3 for the SROIE. In addition, results over edit operations are only presented for the best combination approaches in the comparative Table VIII for FUNSD and Table IX for SROIE. Finally, Table X presents examples of outputs for two sample text images from the SROIE dataset, those outputs are presented for the engine and combinations that achieved the best results for the SROIE.

For the FUNSD dataset, the results surpassed the best individual results more than once. Observing Figure 2, there was an improvement in terms of character accuracy for ten combinations, but only for four in terms of word accuracy. Besides that, comparing the best individual performance to the combinations in Table VIII, it is observed that there is greater improvement in terms of character accuracy than word accuracy. This may not be something necessarily bad because it indicates that the system improved more on correctly identified characters over whole words, which is important for alphanumeric codes that can not be matched with a dictionary.

The best result was achieved with the median string with confidence weight for the combination of three Tesseracts (wei_median_t4p6_t5p6_t5p12) that individually have achieved the best individual performances for the FUNSD dataset. In this experiment for the FUNSD, the best result in terms of character accuracy was 1.48% better than the best individual result with the majority voting combination with all psm values from Tesseract 5 (maj_t5p_all), and 0.54% better in terms of word accuracy with median string weighted (wei_median_t4p6_t5p6_ t5p12). Even though the majority voting achieved greater character accuracy, the weighted median string still surpasses the best individual results for this metric.

For the SROIE, the results surpassed the best individual results more than once as well. Observing Figure 3, there was an improvement for eleven combinations in terms of character accuracy and three combinations for word accuracy. In the same way as the FUNSD, there were greater improvements in terms of character accuracy.

The best result was also observed with the median string

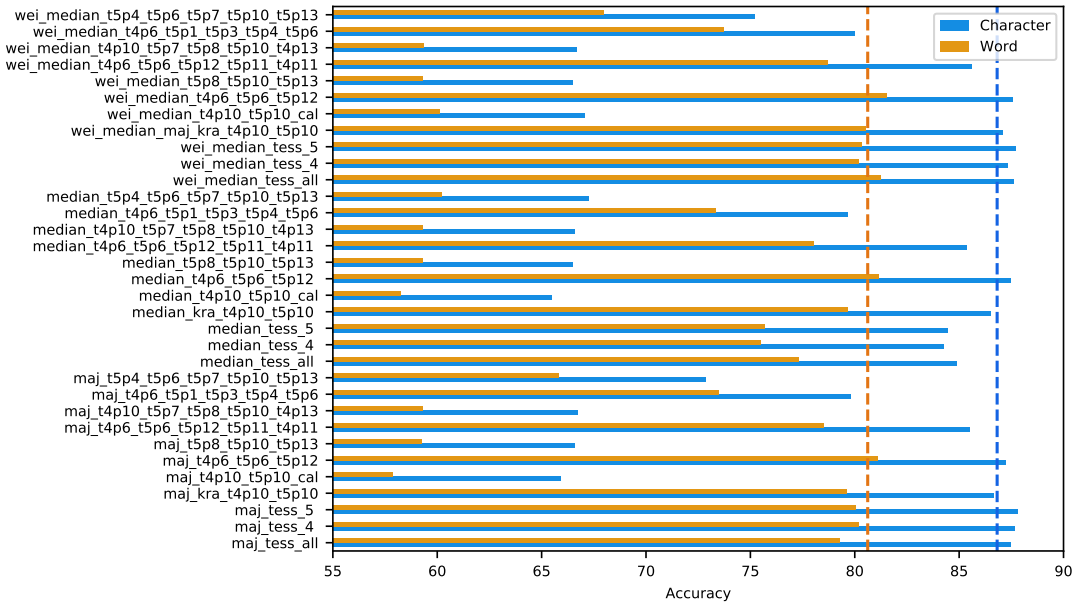


Fig. 2. Accuracies results on FUNSD with *psm* variation, the dashed lines represent the best individual result.

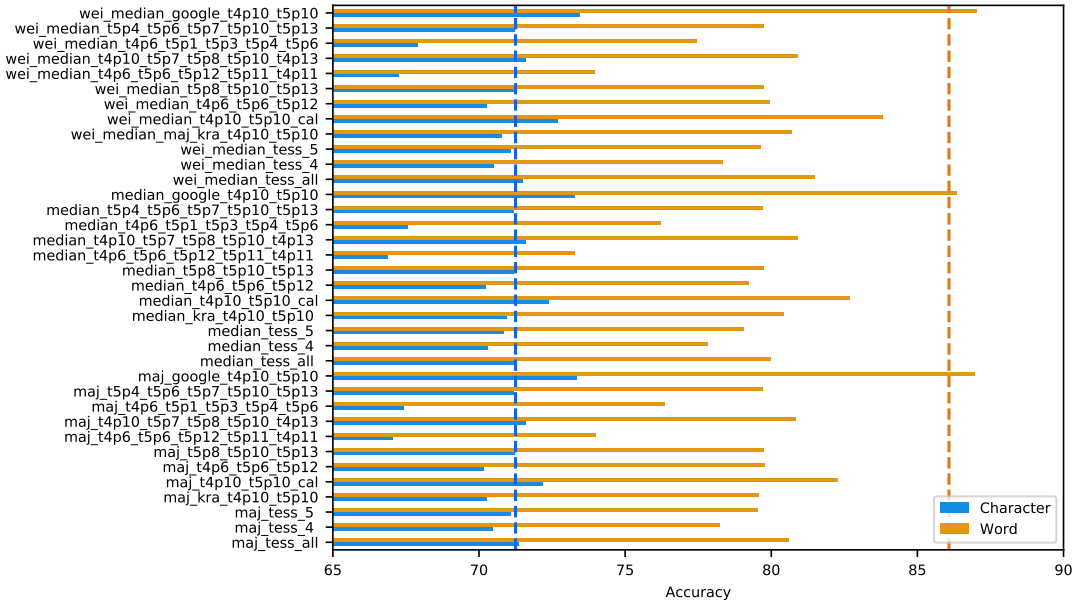


Fig. 3. Accuracies results on SROIE with *psm* variation, the dashed lines represent the best individual result.

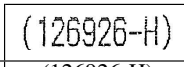
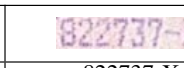
TABLE IX
BEST RESULTS COMPARISON ON SROIE.

	SROIE		
	Individual	Combination with Google Vsion	Combination without Google Vsion
	Google Vision	wei_median_google_t4p10_t5p10	wei_median_t4p10_t5p10_cal
Character accuracy	71.25	**73.45**	72.7
Word accuracy	86.08	**87.08**	83.81
Insertion	175953	**160051**	162439
Deletion	**150593**	159331	160994
Substitution	**14947**	15375	18129

with confidence weight (wei_median_google_t4p10_t5p10), the set of OCR systems was the Google Vision, Tesseract 4.1 $psm = 10$ (sparse text), and Tesseract 5.0 $psm = 10$. This result achieved a character accuracy 3.09% higher in comparison to the Google Vision individual result (best individual result on SROIE), and 1.16% higher for word accuracy.

More interestingly, is that even without using the Google Vision OCR in the combination set, it was possible to surpass its individual character accuracy. This was possible through the combination of two Tesseracts and the

TABLE X
BEST RESULTS COMPARISON ON SROIE.

		
Correct Text	(126926-H)	822737-X
wei_median_google _t4p10_t5p10	(126926-H)	G2e451TK
wei_median_t4p10 _t5p10_cal	(126926-H)	a22731-X
Google Vision	(126926-1)	No text was identified

Calamari with our weighted median string combination (wei_median_t4p10_t5p10_cal). This result is really interesting because Tesseract and Calamari are free, differently from Google Vision. This result is also illustrated in Table IX.

V. CONCLUSION

This work brought a new proposal for the combination of pre-trained models of OCR, such as Tesseract engine or Google Cloud's Vision API, for example. The combination was achieved at the decision level for the characters, i.e. the combination happened after all the models gave their output, and there is no combination in the pre-processing stage or the segmentation step. It was explored three different combination approaches between different sets of the five OCR engines; majority voting, ranking, and median string, which is a concept newly explored in the context of documents with sparse text and alphanumeric codes. For the validation, three different metrics were calculated with two datasets.

Throughout this work, reference works were presented as well as some necessary concepts. In which, the median string concept stands out as an effective method for combining OCR engines and helps to deal with how to compare the different outputs. With the median string, it was achieved an improvement of 3.09% and 1.16% in terms of character accuracy and word accuracy respectively, in the SROIE dataset. In the FUNSD dataset, it was observed an improvement of 1.48% in character accuracy with majority voting and 0.54% in word accuracy with median string which also surpassed the individual best character accuracy. Also, in the SROIE, it is interesting to note that even in combinations without the Google Vision was possible to achieve better results through the median string combination between Tesseract and Calamari.

These results indicate that the combination of the outputs from different OCR engines presents improvement in comparison to the individual results. Also, this work brought the Kohoken [3] median string concept as a combination approach, which yielded the best results for all combination experiments of this work. Finally, this work is interesting as a way to easily achieve some improvement through OCR engines, without training.

For future works, exploring the combination of the outputs from the same document with different preprocessing strategies as proposed by Boiangiu et al. [8] can be interesting, as well as to explore ways to deal with different segmentations. Furthermore, experiments building a deep learning model specifically for this kind of data should also be conducted, and also experiments involving other OCR systems with open code.

REFERENCES

[1] J. Memon, M. Sami, and R. A. Khan, "Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR)," 2020, arXiv:2001.00139.

[2] D. Vishwanath, R. Rahul, G. Sehgal, Swati, A. Chowdhury, M. Sharma, L. Vig, G. Shroff, and A. Srinivasan, "Deep reader: Information extraction from document images via relation extraction and natural language," in *ACCV*, Cham, 2019, pp. 186–201.

[3] T. Kohonen, "Median strings," *Pattern Recognition Letters*, vol. 3, no. 5, p. 309–313, 1985.

[4] M. Mohandes, M. Deriche, and S. O. Aliyu, "Classifiers combination techniques: A comprehensive review," *IEEE Access*, vol. 6, pp. 19 626–19 639, 2018.

[5] O. Petrova, K. Bulatov, and V. Arlazarov, "Methods of weighted combination for text field recognition in a video stream," 2019, arXiv:1911.12028.

[6] P. Singh, A. Verma, and N. S. Chaudhari, "Feature selection based classifier combination approach for handwritten Devanagari numeral recognition," *Sadhana*, vol. 40, pp. 1701–1714, 2015.

[7] M. R. Ahmadzadeh, M. Petron, and K. R. Sasikala, "The Dempster-Shafer combination rule as a tool to classifier combination," in *IGARSS*, vol. 6, 2000, pp. 2429–2431.

[8] C.-A. Boiangiu, R. Ioanitescu, and R.-C. Dragomir, "Voting-based OCR system," *Journal of Information Systems & Operations Management*, vol. 10, no. 2, pp. 470–486, 2016.

[9] R. Petrescu, S. Manolache, C. Boiangiu, C. Avatavului, M. Prodan, I. Bucur, and G. Vlasceanu, "Combining Tesseract and Asprise results to improve OCR text detection accuracy," *Journal of Information Systems & Operations Management*, vol. 13, no. 1, pp. 57–64, 2019.

[10] D. Dannélls, T. Johansson, and L. G. Björk, "Evaluation and refinement of an enhanced ocr process for mass digitisation," in *DHN*, vol. 2364, Copenhage, 2019, pp. 112–123.

[11] C. Reul, U. Springmann, C. Wick, and F. Puppe, "Improving OCR accuracy on early printed books by utilizing cross fold training and voting," in *DAS*, Viena, 2018, pp. 423–428.

[12] L. F. Zeni and C. Jung, "Weakly supervised character detection for license plate recognition," in *SIBGRAPI*, 2020, pp. 218–225.

[13] S. Montazzolli and C. Jung, "Real-time brazilian license plate detection and recognition using deep convolutional neural networks," in *SIBGRAPI*, 2017, pp. 55–62.

[14] L. Han, S. Zou, D. He, and W. J. Zhou, "Detection for mixed-characters based on machine learning," in *Recent Trends in Intelligent Computing, Communication and Devices*, vol. 1006, Singapura, 2020, pp. 167–173.

[15] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: an efficient and accurate scene text detector," 2017, arXiv:1704.03155.

[16] R. Smith, "An overview of the Tesseract OCR engine," in *ICDAR*, vol. 2, Curitiba, 2007, pp. 629–633.

[17] K. Baierer, R. Dong, and C. Neudecker, "Okralact - a multi-engine open source OCR training system," in *HIP*, Nova Iorque, 2019, pp. 25–30.

[18] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 418–435, 1992.

[19] K. Katoh, K. Misawa, K. K-i, and M. T, "MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform," *Nucleic Acids Research*, vol. 30, no. 14, p. 3059–3066, 2002.

[20] G. Jaume, H. K. Ekenel, and J. Thiran, "FUNSD: A dataset for form understanding in noisy scanned documents," in *ICDARW*, vol. 2, Sydney, 2019, pp. 1–6.

[21] Z. Huang, K. Chen, J. He, X. Bai, D. Karatzas, S. Lu, and C. V. Jawahar, "ICDAR2019 competition on scanned receipt OCR and information extraction," in *ICDAR*, 2019, pp. 1516–1520.

[22] S. Rice and T. Nartker, *The ISRI Analytic Tools for OCR Evaluation Version 5.1*, Information Science Research Institute, 1996.

[23] S. Rice, F. Jenkins, and T. Nartker, "The fifth annual test of OCR accuracy," Information Science Research Institute, Las Vegas, Tech. Rep., 2012.