An Optical Character Recognition Post-processing Method for technical documents

Lucas Viana da Silva*, Paulo Lilles Jorge Drews Junior† and Sílvia Silva da Costa Botelho‡
*Universidade Federal do Rio Grande (FURG)
Email: lucas.viana.rs@gmail.com
†Universidade Federal do Rio Grande (FURG)
Email: paulodrews@furg.br
‡Universidade Federal do Rio Grande (FURG)
Email: silviacb@furg.br

***Abstract*—Methods for correcting errors generated by Optical Character Recognition (OCR) system are being developed for a long time, with interesting results in their applications. However, these methods tend to work only on data with words that are part of an existing language and with a large semantic relationship between each word in the text. In this work, an error correction method is proposed that focuses on types of documents without these large semantic relationships inside their text. Instead, we focus on sparse text that tends to have little semantic relationship between the words found within itself. The proposed method uses machine learning to train classifiers capable of finding errors in the OCR output and run an isolated execution of the OCR system to fix the error. The final results indicate a good accuracy of 88.24% for error detection and an improvement of the character error rate (CER) of 14.2%.**

I. Introduction

OCR is a tool for transforming text documents from an image to text, which helps with the manipulation and searching of data inside a document. This technology is becoming more and more important, mainly due to the improvement of the results obtained with modern OCR systems. Nowadays, they reach a high accuracy rate in most circumstances because of the use of deep learning. However, the OCR usually produces several errors in the recognized texts depending on the type of document to be recognized, making it necessary to have some way of post-processing it to recognize and correct these errors.

In the context of general OCR post-processing methods, there are several studies [1], [2] that use word dictionaries and context to detect and correct the text. The use of dictionaries and context shows a good result for detecting and correcting errors in texts taken from books and articles, where the detected words are related depending on the proximity between each one and most of these words could be found in a dictionary of the language it was written. The problem with them is their insufficient performance when used for documents with little information by context and/or with difficulty to create a dictionary of words to detect similarity between the estimated text and the words in the dictionary. An example of this situation is found when trying to correct errors in a technical document. Technical documentation usually describes the application, purpose, creation, or architecture of a product or service. They are composed of several snippets such as codes, item prices, and drawings. These items tend to be unique to the document and cannot be found in a dictionary, thus there is normally no concrete context to predict them. Thus, there is a greater chance of error in the found text in modern OCR systems such as Tesseract [3], because most of these systems utilize attention-like mechanisms such as Long short-term memory (LSTM) [4] and Transformers [5] as a character recognition model, which is influenced by the context that the words being recognized are in. With that in mind, this article's main objective is to be able to classify and fix OCR errors that occur when trying to utilize it on technical documents which are sparse and without significant language context between many of its words. This method will do that through a simple pipeline seen in Figure 1, common among all OCR post-processing methods, utilizing techniques such as feature extraction and classification for error detection and a special execution of an OCR system for error correction.

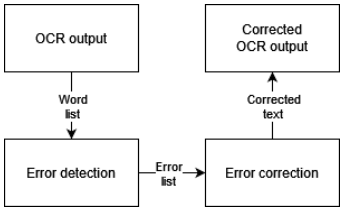


Fig. 1. The pipeline of the OCR post-processing method.

II. Related Works

Several works present techniques for the identification and correction of errors in OCR systems, where the majority fits into one or more within three broad categories.

Manual approaches utilize human interaction with a system to identify and correct OCR errors [6], normally making use of crowd-sourcing to facilitate and increase the number of corrected terms. The obvious drawback of these techniques is the fact that they have to be manually executed by a human which contradicts the usage of OCR systems as a way to automate processes.

Context-dependent approaches use the context of each word to try and determine if the element being analyzed is an error and which text should be in its place. The most common techniques from this category are statistical language models [7] which estimate the probability distribution of word sequences; and neural network-based language models [2] which utilizes a neural network to model what would amount to the probability distribution of word sequences. Other types of context-dependent techniques include sequence-to-sequence models [8], which interpret the problem of correcting OCR errors as if it were a machine translation problem, *i.e.* translating from an incorrect "language" to one that should be correct. These methods do not tend to fit the problem of OCR post-processing in technical documents, mostly because it is difficult to derive context from them.

Feature-based approaches utilize a machine learning model to try and learn which words in an OCR output are wrong and which are correct, mostly through classifiers that use techniques such as k-nearest neighbor (KNN) and support vector machines (SVMs). Most of them tend to utilize features such as n-gram frequencies [1] to try and include some context within the selected features, and OCR word confidence [9] to include the uncertainty of the OCR model. Particular word/character feature selections are also adopted such as the presence of non-alphanumeric characters [10] to detect anomalies in OCR outputs of documents that normally should not have them, and the edit distance from the analyzed word to candidates in a dictionary [10] to include some semblance of language-based techniques. Some implementations also utilize multiple OCR outputs, merging them together [11] to take advantage of the strengths of each specific OCR to detect and recognize texts in different situations. This category has the most in common with the proposed method, but they differ because the method does not include features that have any correlation with context or a language model. Instead, we focus purely on the word and character uncertainty features with the expected result of finding and correcting these errors even in a document of sparse text and limited context.

III. Methodology

In Figure 1 we have the main pipeline of an OCR post-processing method. This paper follows that pipeline by extracting features from both the OCR output and the image of each word in the document and classifying it into a correct or incorrect word, with the classification methods that will be presented in a later section. After classification, a list of errors will be generated from the input words, where each word will be cropped from the original image and then reprocessed in the OCR system with a specific configuration and some image preprocessing techniques to reduce the error compared to the original output. In the next few subsections the topic of datasets utilized for training the classification models, the features utilized for said models, and the specific configuration of both the OCR and image manipulation techniques utilized during error correction.

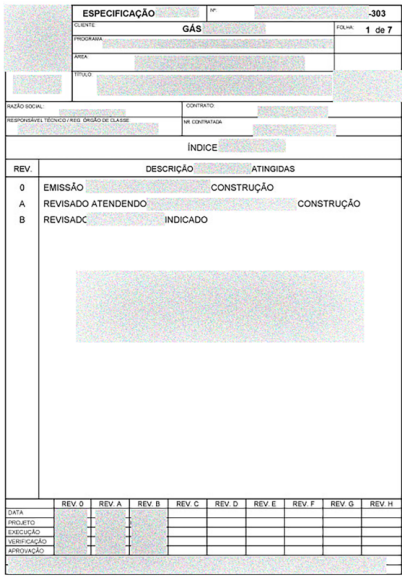
Fig. 2. An example of an image from the SROIE dataset [12].

A. Dataset

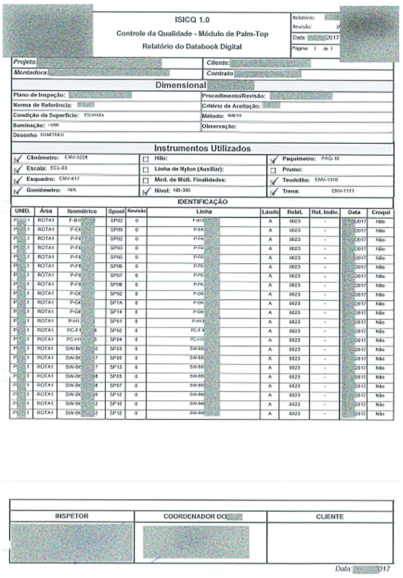
Two datasets are adopted in this paper. The first is SROIE [12], a largely adopted public dataset of scanned receipts that contains text information that is sparse in its layout when compared to other documents such as books, while also being hard to derive context from said images due to the amount of numbers, unique names, and codes that are contained within them. Figure 2 shows an example of a scanned receipt in the dataset. Each image in the dataset was then put through an OCR system to be able to correlate the ground truth text with the OCR-obtained text. The dataset contains $1,000$ images of scanned receipts, with the text inside those images mainly consisting of digits and English characters, with some symbols in places such as dashes in codes and dots in prices of items.

The second one is a private dataset [13] called ENAVAL consisting of scanned technical engineering documents related to normative, construction, and assembly processes of the naval and offshore industry. This dataset consists of 5720 pages with two types of documents, one related to a technical normative of an engineering project and the other type related to materials used during the execution of these projects. Figures 3a and 3b demonstrate an example of both types of documents contained within this dataset.

The OCR system utilized in this paper was the Tesseract [3], the most adopted open-source OCR system which has a LSTM model as its recognition backbone. The Tesseract was customized for the task of extracting text in a sparse document, changing default parameters to others such as: *tessedit_pageseg_mode* being set to 11, the trained model of Tesseract varies for the dataset being analyzed, with English being used for SROIE and Portuguese being used for ENAVAL, and two parameters to have more information about the OCR output. These two parameters are *hocr_char_boxes* equal to 1 that outputs the bounding boxes and certainty for each character instead of just the words, and



(a) Example of the first type of document in the ENAVAL dataset.



(b) Example of the second type of document in the ENAVAL dataset.

Fig. 3. An example of documents present in the ENAVAL dataset.

lstm_choice_mode set to 2 outputting which other characters outside of the one chosen were considered for each character in a word. All visualizations such as histograms shown in Section III-B are done utilizing the SROIE dataset with $1,000$ correctly identified words and $1,000$ incorrectly identified words of this dataset. Datasets were divided into 85% of the data used to train, and the remaining 15% being used to test.

B. Error Detection

Five different features are utilized within machine learning classifiers, which are detailed in the next five subsections. We adopted five machine-learning algorithms to obtain classify errors:

- k-Nearest Neighbours (KNN) with 3 neighbors;
- Support Vector Machine (SVM);
- Naïve-Bayes (NB);
- Decision Tree (DT);
- Multilayer Perceptron (MLP).

These classifiers were chosen due to the proof of their effectiveness in other applications within machine learning and also to guarantee a range of different classifiers for comprehensive evaluation using various techniques in the field of machine learning. The use of large deep learning models is discarded due to the limited amount of annotated data and the high cost to obtain larger datasets. All classifiers were made utilizing the library Scikit-Learn [14]. All data were preprocessed by standardizing every feature before the training and testing of any classifier. For every feature value 'x' the Equation 1 is applied, where 'u' is the mean of the training samples for that particular feature and 's' standard deviation of the training samples for that particular feature.

$$Standard\ Value = (x - u)/s \tag{1}$$

We paired up the ground truth text and the OCR-based generated text using an Intersection-over-Union (IoU) between the bounding boxes of the words inside the OCR output and the words inside the ground truth, pairing up any words with more than 0.5 IoU.

One experiment made was the training of the classifiers with all combinations of the five features that will be described in the later subsections. This was made to be able to ascertain that the combination of all five features at the same time was the best in terms of performance for error detection. Another experiment was made to see the impact of Bagging [15] and Boosting(AdaBoost) [16] techniques in these classifiers, with the specific notion that, due to limitations in the library utilized to make these classifiers, only the Decision Tree classifier and the SVM classifier where able to be utilized with Boosting. The experiment utilized boosting in all types of classifiers.

1) OCR Word confidence: The simplest feature is the confidence that Tesseract has for a given word. This value is a number from 0 to 100, with the lower bound indicating total uncertainty and the upper bound indicating no uncertainty by Tesseract. Figure 4a shows a histogram that makes it possible to see that, though incorrect words tend to have lower values, it is still impossible to judge a system completely through this feature alone.

2) Average confidence of characters: This feature represents the average confidence that Tesseract has between all character confidences. This feature is not part of the default metrics in Tesseract and is constructed after the output of the OCR is obtained. This value is a number from 0 to 100, with the lower bound indicating total uncertainty and the

upper bound indicating no uncertainty by Tesseract. Figure 4b shows a histogram that, much the same as the word confidence feature, makes it possible to see that, though incorrect words tend to have lower values, it is still impossible to judge a system completely through this feature alone.

3) Confusion value of word: Feature generated after processing the OCR output, describing the sum of the values of character confusion in the OCR system, which originate from a previous analysis, comparing the data obtained through OCR with the annotated dataset to obtain the number of errors for each character. To create this value, the Levenshtein distance is used to find how many times each distinct character was replaced with another character in the pairs between OCR words and the annotated text in the database. This information is used to calculate the confusion value of each character and the total value of the characteristic, with the sum of all confusion values of each character in a word being the final value of this feature. This value has 0 as the lower bound which characterizes no character confusion and any higher value characterizes more confusion for that word. Figure 5a shows a histogram that, different from the last two, does not show an immediate clear pattern for classification, but has some subtle differences for correct and incorrect words in a few ranges that help the classification on an error.

4) Average of choice difference: Feature generated after processing OCR output, describing the average of all OCR choice differences in a word for each character between its chosen character and the second highest character possibility. A choice difference for a character is the difference between the highest-confidence candidate minus the second-highest-confidence candidate for that character. This value is a number from 0 to 100, with the lower bound indicating total uncertainty and the upper bound indicating no uncertainty by Tesseract. Figure 5b shows a histogram that, much the same as the word confidence feature, makes it possible to see that, though incorrect words tend to have lower values, it is still impossible to judge a system completely through this feature alone.

5) Character Gradient Image Quality Assessment: Feature generated after processing OCR output, derived from a technique to assess the quality of an image for an OCR process [17], being applied to the image of the word found in the OCR that was paired with the text in the ground truth. This value is a number that indicates the quality of an image in relation to its supposed OCR result, with lower values indicating worse quality. Figure 6 shows a histogram that shows some differences in words correctly found and incorrectly found, especially at the lower bound values, which does make sense as they represent a worse quality image and they tend to be recognized incorrectly by Tesseract.

C. Error Correction

An isolated execution of the Tesseract OCR system was applied to correct errors. We cropped the image to a candidate wrong word and applied the OCR with preprocessing and internal configurations designed to work better with isolated text images. The preprocessing involves the resizing of the image to be 35% larger, with a bilinear interpolation of new pixels, followed by the application of unsharp masking with a size 3 kernel of a Gaussian blur to sharpen the image. The internal configurations of OCR are as follows:

- Page segmentation mode: 8 (Single Word);
- OCR engine: Legacy (utilizing a KNN model as the recognition model);
- Language model: English;
- Internal thresholding method: Sauvola;
- kFactor (Sauvola parameter): 0.1;
- Window Size (Sauvola parameter): 0.4.

To compare how the OCR behaves before and after post-processing the Character Error Rate (CER) metric was utilized. This metric determines how much error is in the OCR output by comparing its output with the ground truth through an edit distance technique, which in this case is the Levenshtein Distance. The formula for the CER is described in Equation 2. A higher CER indicates a worse OCR output.

$$CER = \frac{(Substitutions + Deletions + Insertions)}{(Number\ of\ characters\ in\ ground - truth)} \tag{2}$$

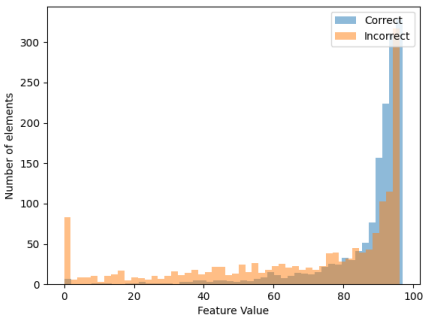
Two experiments with error correction were made, with the first utilizing the classifier with the best accuracy and then, for each word classified as an error, an isolated OCR execution was made. The second experiment involved bypassing the error detection of the method to check if applying an isolated OCR execution to all words in the image was better than checking for errors before. This choice was made because while a specific OCR execution can help reduce the error in wrong words, it can also introduce errors to words that were correct in the first place, so this experiment was conducted to measure how vital error detection is to the entire process.

IV. Results

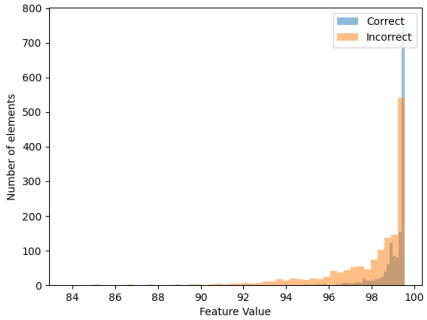
A. Error Detection

Table I shows the best classifier, ranked by accuracy, for each combination of features, with a combination being shown as the sum of item indexes above for each feature in Section III-B. For example, the combination "(2+3+4)" is equal to training the classifiers only with the average character confidence, confusion value, and the average of choice difference. This experiment was performed only on the SROIE dataset. In the table, only the best five out of the thirty-one combinations are shown. The results in this table indicate that, though by a small margin, utilizing all five features is the best choice for the classifier, with all of the next experiments being done with all five.

Table II show the performance for all the classifiers with and without the usage of Boosting and Bagging in the SROIE dataset. It can be seen from the results that Bagging and Boosting do not seem to have a great effect on most classifiers, with the exception being the Decision Tree classifier which ends up being the most accurate with Boosting and second

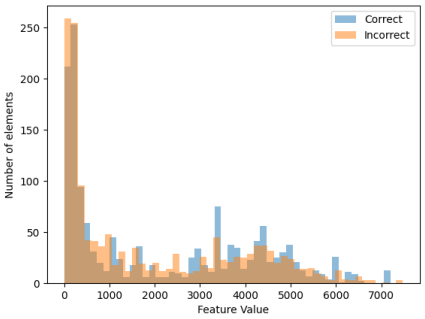


(a) A histogram that shows the contrast of the word confidence feature between words that were found correctly and incorrectly.

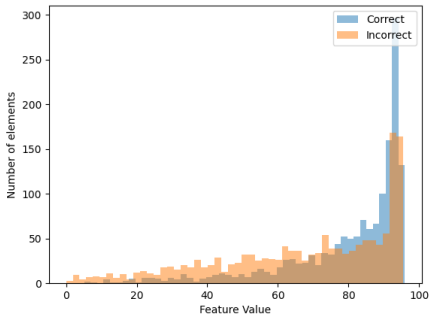


(b) A histogram that shows the contrast of the average character confidence feature between words that were found correctly and incorrectly.

Fig. 4. Features that come from the confidence value of elements in the OCR-generated text.



(a) A histogram that shows the contrast of the confusion value feature between words that were found correctly and incorrectly.



(b) A histogram that shows the contrast of the average of choice difference feature between words that were found correctly and incorrectly.

Fig. 5. Features that are generated after processing the OCR output.

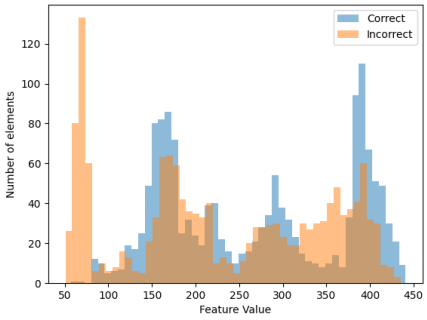


Fig. 6. A histogram that shows the contrast of the Character Gradient Image Quality Assessment feature between words that ere found correctly and incorrectly.

in all the other metrics. It can also be seen that recall and precision are a bit low across the board, which could introduce false positive OCR errors that can impact the step of error correction, being analyzed in the next subsection.

Table III shows the performance for all the classifiers in the ENAVAL dataset. It can be seen that the KNN is the clear winner among all classifiers, with the Decision Tree in second place. It can be seen here too that recall and precision are a bit low across the board, even more than the SROIE dataset, which could introduce false positive OCR errors that can impact the step of error correction, being analyzed in the next subsection.

TABLE I
RESULTS OF THE BEST CLASSIFIER FOR EACH OF THE 5 BEST COMBINATIONS IN THE SROIE DATASET

Combination	Accuracy	Classifier
(1+2+3+4+5)	86.59%	MLP
(1+3)	86.14%	Decision-Tree
(2+5)	86.08%	MLP
(2+4)	86.07%	KNN
(1+2+4)	86.02%	SVM

B. Error Correction

Table IV shows the end result of error correction, where ”Full Pipeline” means that both error detection, with the best

TABLE II
RESULTS OF EACH CLASSIFIER IN THE SROIE DATASET

Classifier	Accuracy	Precision	Recall	F1 Score
Naïve-Bayes (NB)	85.78%	66.79%	54.84%	60.23%
NB - Bagging	85.91%	67.83%	54.73%	60.58%
KNN	86.37%	67.69%	58.50%	62.76%
KNN - Bagging	85.69%	67.23%	55.67%	60.91%
SVM	86.52%	71.42%	44.64%	54.94%
SVM - Bagging	87.14%	74.76%	48.94%	59.16%
SVM - Boosting	84.63%	69.21%	53.19%	60.15%
Decision Tree (DT)	86.21%	64.77%	**65.23%**	65.00%
DT - Bagging	88.16%	**76.82%**	58.51%	**66.43%**
DT - Boosting	**88.24%**	74.96%	59.88%	66.33%
MLP	86.96%	73.38%	52.72%	61.35%
MLP - Bagging	85.89%	71.25%	49.48%	58.40%

TABLE III
RESULTS OF EACH CLASSIFIER IN THE ENAVAL DATASET.

Classifier	Accuracy	Precision	Recall	F1 Score
Decision Tree	81,21%	**58,16%**	63,58%	60,75%
Naïve-Bayes	75,12%	34,01%	50,35%	40,60%
KNN	**82,31%**	57,56%	**67,03%**	**61,94%**
MLP	77,88%	33,14%	60,51%	42,82%
SVM	78,07%	32,01%	61,88%	42,20%

TABLE IV
RESULTS OF THE ERROR CORRECTION TESTS.

Dataset	CER Original	CER Full Pipeline	CER Only Error Correction
SROIE	0.4473	0.3838	0.7224
ENAVAL	0.7710	0.7349	0.8612

classifier by accuracy for each dataset, and error correction were used, and "Only Error Correction" means that all words were processed with the error correction method, without the classification into error or not. It is important to note that, even through reprocessing of false positives, the error rate after post-processing is lowered by 14.2% in the SROIE dataset, a significant increase that could benefit other activities that depend on the OCR results such as text searching or tasks of structured information retrieval, with a common example being layout and forms understanding, a task which is quite common in technical documents such as the ones discussed on this paper. It is also possible to see that without the step of error detection, the CER gets worse, which is not too surprising considering our hypothesis in Section III-C, which was that reprocessing words that were correct after the first OCR pass could introduce errors and, in this case, could become worse than the original output. The results in the ENAVAL dataset also present the same pattern, with a 4.7% lowered error rate for the full processing and an increase of 11.7% in the error rate if only using the error correction step.

V. CONCLUSION

In this work, we analyzed and presented a method to find and reduce OCR errors in technical documents using light classification methods such as KNN, SVM, and Decision Trees for the identification of incorrect words, and the isolated execution of OCR for the reduction of errors in said words. The obtained results improved the performance by 14%. To conclude this research we still need to evaluate certain aspects such as the relatively low scores in error classification, possibly caused by an unbalanced dataset, and compare it to different post-processing methods to accurately measure this technique. Furthermore, there is a necessity to see the behavior of this technique while utilizing other OCR systems, to be able to conclude if this can be generalized or if it is a technique that only works in specific situations.

REFERENCES

[1] J. Mei, A. Islam, Y. Wu, A. Moh'd, and E. E. Milios, "Statistical learning for ocr text correction," *arXiv preprint arXiv:1611.06950*, 2016.

[2] E. D'hondt, C. Grouin, and B. Grau, "Low-resource ocr error detection and correction in french clinical texts," in *Proceedings of the seventh international workshop on health text mining and information analysis*, 2016, pp. 61–68.

[3] R. Smith, "An overview of the tesseract ocr engine," in *Ninth international conference on document analysis and recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 629–633.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[6] R. Holley, *Many hands make light work: Public collaborative OCR text correction in Australian historic newspapers*. National Library of Australia, 2009.

[7] A. Poncelas, M. Aboomar, J. Buts, J. Hadley, and A. Way, "A tool for facilitating ocr postediting in historical documents," *arXiv preprint arXiv:2004.11471*, 2020.

[8] V. Nastase and J. Hitschler, "Correction of ocr word segmentation errors in articles from the acl collection through neural machine translation methods," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[9] I. Kissos and N. Dershowitz, "Ocr error correction using character correction and feature-based word classification," in *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. IEEE, 2016, pp. 198–203.

[10] G. Khirbat, "Ocr post-processing text correction using simulated annealing (opteca)," in *Proceedings of the Australasian Language Technology Association Workshop 2017*, 2017, pp. 119–123.

[11] I. L. Correa, P. L. J. Drews, and R. N. Rodrigues, "Combination of optical character recognition engines for documents containing sparse text and alphanumeric codes," in *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2021, pp. 299–306.

[12] Z. Huang, K. Chen, J. He, X. Bai, D. Karatzas, S. Lu, and C. Jawahar, "Icdar2019 competition on scanned receipt ocr and information extraction," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 1516–1520.

[13] G. L. Santos, V. T. Silva, L. A. Dalmolin, R. N. Rodrigues, P. L. Drews, and N. L. Duarte Filho, "A form understanding approach to printed and structured engineering documentation," in *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2021, pp. 330–337.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[15] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[16] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[17] H. Li, F. Zhu, and J. Qiu, "Cg-diqa: no-reference document image quality assessment based on character gradient," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 3622–3626.