Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 207 (2022) 1571–1580

Procedia Computer Science

www.elsevier.com/locate/procedia

26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2022)

Official Document Identification and Data Extraction using Templates and OCR

Cosmin Irimia, Florin Harbuzariu, Ionut Hazi, Adrian Iftene

"Alexandru Ioan Cuza" University of Iasi, Faculty of Computer Science

General Henri Mathias Berthelot Street, No. 16, 700259, Iasi, Romania

Abstract

Nowadays anyone possesses at least one personal document, whether it is an identity card or a driving license. In many of our daily activities, working with documents or the necessity to present them in different contexts became something normal. Given the current situation, we are all going through, having to deal with the current pandemic situation, we have adapted to some extent to new ways of communicating, working, and resolving things, in general, using Internet tools and platforms. In this context, a small part of the fight we have with the bureaucracy has been won, many of the documents that once could only be brought only in physical locations can now be scanned and sent by email or attached to a form on the website. Hoping that the pandemic has also taught us good things, we want to believe that these ways of communication will be kept in place even after this issue is over, to avoid unnecessary waste of time and congestion. The focus of this project is developing an application that can create textual information data from a simple image provided by the user that will enable the possibility of sharing digital versions of documents, making our lives so much better.

Keywords: Computer vision; Algorithm optimization; Image processing; Optical Character Recognition;

1. Introduction

Nowadays, obtaining official documents from local authorities can often be a daunting task that can lead to many trips to specialized offices and many hours lost in traffic and counters. Just imagine that in the attempt to issue a criminal record: at this moment the citizen must make an appointment, go to the police station that deals with the

* Cosmin Irimia. Tel.: +4-0232-201549;
E-mail address: cosmin.irimia@info.uaic.ro

1877-0509
10.1016/j.procs.2022.09.214

release of the criminal record, and fill out a form with the data from the identity card, present the form and the identity card for comparison, then the police officer must enter this data which was handwritten into the computer and the system will then generate the criminal record, the police officer will print it and after so much time wasted, the citizen finally gets the desired document. This operation can be drastically simplified by using an automated system that can receive requests with attached documents, scan, identify, and obtain relevant information from those files for further use or correlation with other datasets or systems would be crucial and would make bureaucratic processes much simpler and faster.

If we had such a system, the process would look something like this: the citizen scans the identity card, sends it to the email of the police institution with the subject ”criminal record” and based on an email rule the system will download the attached document, will extract the personal data from it and it will automatically generate the criminal record, sending it to the email of the person who initiated the request. Of course, with such a system, certain security rules must come to ensure data privacy and GDPR, but these can be established through a protocol, not influencing the way the system manages the documents. In this sense, we want to propose a system for identifying the type and content of any document to streamline this process. Even if we can convince all of the government bodies to accept our documents in all-digital [12] format, it's still very tough to go through all these documents and a person will still have to manually look at all the scanned emails and documents to receive and index all the necessary data to obtain that piece of information the citizen wants; This process is difficult and expensive both in terms of time and money.

In this context, our research proposes a solution that it's able to retrieve information from official documents. It's not just very fast, it's very precise, efficient, and accurate as well. The main research questions of this paper intend to answer: (1) *How does image resizing affect processing times of identification algorithms*? (2) *What is the best approach to enhance images to be at their best quality for performing OCR operations*?

The paper is structured as follows: Section 1 introduces us to the problem where we will explain why we need such a system and how it helps us. Section 2 presents a brief overview of current information extraction systems to clarify their importance and what we can do to add new system requirements, while Section 3 presents our solution, its architecture, and algorithms. Section 4 presents our initial results and a partial conclusion, all the iterations that we‘ve been through, the improvements and all the modifications that we‘ve done to obtain better results and the last 2 sections talk about our future work and conclusions.

2. Background

The first OCR [16] device was developed by Gustav Tauschek, an Austrian engineer, in the 1920s. He obtained a patent on his so-called “Reading Machine” in Germany in 1929. The second person who obtained a patent on OCR was Paul Handel with his “Statical Machine” in the USA in 1933. Early versions of OCR were limited to recognizing one font at a time. In the 1970s, Ray Kurzweil commercialized “omni-font OCR” [11], which could process text printed in any font. While early OCR [6] techniques worked one font at a time and needed to be trained with images of each character, today's systems are capable of recognizing most fonts with a high degree of accuracy [2]. OCR solutions often preprocess images to improve the chances of successful recognition, using techniques like deskewing (making lines of text perfectly horizontal or vertical), binarisation (converting an image from color or grayscale [9] to black-and-white to efficiently separate the text from the background), line removal (cleaning up non-glyph boxes and lines) zoning (identifying columns, paragraphs, etc.), line and word detection [24] (setting a baseline for word and character shapes, separating words if needed), etc.

Over time, many governments and companies have developed similar systems to perform this data extraction, each with its quirks and features. Before talking about the existing solutions we must mention the fact that no such system can scan any type of document, they all are built to work by regions or states, we have not found a generally valid and fully working solution.

2.1. Sinosecu

Sinosecu is an OCR recognition product developed by a Chinese company to take advantage of cutting-edge OCR recognition technology [19] and it supports structural recognition of all fields on a Chinese ID Card. Their OCR SDK can be used in multiple ways, namely, privatized deployment, mobile SDK, etc. Besides ID Cards, this OCR service

can recognize multiple very well known documents from China such as: *Passport, Bank Card, Marriage Certificate, Military Officer Certificate, Residence Permit, Lawyer's License, Driver's License, Household Register, China Exit and Entry Permit, Premises Permit* and a standardized version of Business Card. Besides Chinese documents, Sinosecu has support for a few international ID Cards such as documents from: Thailand, Singapore, Indonesia, Malaysia, Egypt, Japan, Mexico, and United Arab Emirates. This product has many features such as full-field recognition that supports recognition and output of full-field information (*name, gender, nationality, date of birth, address, ID number*, etc.), it also has edge detection, supports automatic rotation, inclination correction, it supports local recognition, outputs in JSON or XML and also provides an All-in-One SDK that is easily deployable.

From the product page, we can read a great number of features that look very good at a first glance but compared to our solution, Sinosecu can only rotate images if they are in only 3 very good established angles (90°, 180°, and 270°), where our solution can rotate from only 1° up to 359°. Also, this product does not enhance the images judging by the photo lighting conditions looking for the best version of it, it only applies some filters on each and everyone using a "one size fits all" approach that surely will not work on all scenarios. Also, even if the Chinese solution has more documents in its portfolio, it's not horizontally scalable such as our solution that allows adding as many document templates as desired. One last difference is that even if Sinosecu has an offline mode, it's still not open-source, thus making it not as trustworthy as our application because let's now forget that we are using them for handling private and personal information.

2.2. Huawei Cloud Passport OCR

Passport OCR recognizes the text on the first page of a passport and returns the structured information in JSON format. In the current version, all fields of a Chinese passport can be recognized. For non-Chinese passports, two lines of internationally standardized machine-readable codes on the bottom of each passport can be recognized, and 6 to 7 key fields can be extracted from the codes.

Passport OCR has a more general approach to the problem, it can extract information from many ID Cards [23] and Passports from different countries based on the machine-readable code at the bottom of the first page. Also, it can rotate images to any degree unlike Sinosecu's solution described above. However, despite having multiple degrees adjustment, it lacks a ton of features like image enhancement of the low-light photos, cannot auto-scale the images based on their ratios having trouble with images distorted more than 10% of the original aspect ratio. It also lacks offline mode or any type of disconnection from Huawei's servers to ensure the privacy and security of our data. Also, it is not adjustable to the needs, you can not add templates for your specialized needs and if we compare it to Sinosecu or our solution, it has too few document types, to begin with.

3. Proposed Solution

Given all the things we have observed and learned from the solutions presented above, we have proposed and modeled the solution with some well-established principles in mind. Firstly, we consider that this product works directly with personal documents that contain confidential information and this must be as transparent as possible, therefore, the code of our solution is open-source. Secondly, besides the fact that it is open-source, our solution is offline so that anyone can download it on their computer, and a few clicks away there is a product that does not require an Internet connection and that makes it trustworthy. Then, our solution must be easily extensible, so we rely on well-established templates and depending on the need of the use-case and the multitude of documents needed for that occasion [1]. If you want to add a new document type, the system needs a single template of that document and then can determine whether or not the user is scanning that document and it's able to extract the data from it. Last but not least, we want to be able to use the system on less successful images, so we will apply various preprocessing [13] techniques before running the actual OCR [12] to maximize the chances of success in detecting and extracting text.

3.1. OCR Algorithms

While early OCR techniques worked one font at a time and needed to be trained with images of each character, today's systems are capable of recognizing most fonts with a high degree of accuracy. OCR solutions often pre-process images to improve the chances of successful recognition, using techniques like deskewing (making lines of

text perfectly horizontal or vertical), binarisation (converting an image from color or grayscale to black-and-white to efficiently separate the text from the background), line removal (cleaning up non-glyph boxes and lines) zoning (identifying columns, paragraphs, etc.), line and word detection (setting a baseline for word and character shapes [7], separating words if needed), etc. The core OCR algorithms can be put under two main types:

- **Matrix matching** - the image is compared to a stored glyph on a pixel-by-pixel basis. This type works best with typewritten text and not very well when encountering new fonts, it was especially used in older OCR solutions.
- **Feature extraction** - detected glyphs are decomposed into "features" [15, 5] (lines, closed loops, line directions, line intersections, etc.), reducing the dimensionality of the representation and making the recognition computationally efficient. This type is also suitable for handwriting recognition and is found in most modern OCR software.

Modern OCRs also use neural networks intensively (e.g. OCRopus, Tesseract [20]), these being trained, for example, to recognize whole lines of text instead of single characters. There are various methods to evaluate the performance of OCR solutions. A simple way to evaluate the prediction output is with the accuracy metric (indicates a "match" or a "no-match"), but it alone can't assess OCR performance effectively. Instead, error rates are frequently used to determine the extent to which the OCR output text and the reference / "ground truth" text differ from each other. Two important ones are **CER (Character Error Rate)** that is based on the Levenshtein distance, and counts the minimum number of character-level operations needed to transform the reference text into the OCR output. It is better suited for use-cases involving the transcription of particular sequences (IDs, phone numbers, etc.) and **WER (Word Error Rate)** [10] that use the same formula as CER, but operate at the word level instead. It is more applicable for transcriptions of paragraphs and sentences of words with meaning (pages of books, newspapers, etc.).

3.2. Identify the Type of Document

One of the many problems we encountered was that we needed a way to measure the similarity [21] between two images so that we could detect what type of document the user uploaded. That similarity is easier to measure if the two images differ in certain pixels only. For example, it is easier to measure the similarity between a blurred image and the original. There are many ways to compute a number that represent the similarity. We found through trial and error that the best way to calculate the similarity score is the RASE and SSIM methods. RASE, or the Relative Average Spectral Error, is used to evaluate the average performance of image fusion methods for each spectral band. We use RASE during the processing stage of the uploaded document. SSIM, or the Structural Similarity Index Measure, is a method for predicting the perceived quality of digital [4] television and cinematic pictures, as well as other kinds of digital images and videos. SSIM is used for measuring the similarity between two images. The SSIM index is a full reference metric; in other words, the measurement or prediction of image quality is based on an initial uncompressed or distortion-free image as a reference. This method is used at the end of the processing stage.

Fig. 1. Driving license example

For our use case, each document has a defined template which is an image that is loaded only once when the server is started. The template is a random document that was obtained through Google Images. The template's relevant fields are all filled with white space. By doing this, an image can be easier compared to the template because the

fields' value and even the photograph won't negatively affect the computing of the similarity score. Considering that a document may contain multiple pages or sides, each template type has a list of defined pages. All pages have the elements that were described previously (a list of hardcoded coordinates and a list of regex for each field). Each page is also represented by a template image. When the type of the uploaded document is detected, its page is also detected. In this way, the application is not limited only to one side of a document and it increases its usability in daily activities.

Fig. 1 represents a driving license with its 2 sides. As we can see, each side has relevant information that can be extracted using OCR tools. The processing stage consists of three steps:

(1) Finding the contours in an image;
(2) Detecting the one with the biggest area;
(3) Warping it to a new perspective so that it is perfectly aligned with the template.

The problem is that after aligning the image, we need a way to see how much it looks like the current template. Therefore, we use the methods RASE and SSIM. RASE is used to find out how much that currently selected area is similar to the document templates. This method is only used during the processing stage. For different variables in an interval, we obtain an array of multiple contours that is different each time. Because of this, RASE is needed to select the best possible area. RASE is also reliable enough for comparison and faster than SSIM which made it more suitable for this specific problem.

At the end of the processing stage, we compare the altered section with the template using the SSIM method. If it is lower than a hardcoded minimum allowed score, then we know for sure that the extracted section isn't the current document we compared it with. SSIM can compare two images better than RASE but because of its time-consuming disadvantage, we only use it once at the end instead of during the processing stage like RASE.

3.3. Pseudo-Code Algorithm

For a better understanding of our product, we will present the pseudocode algorithm in the listing below:

```
best_match_score <- 0
best_match <- null_object

for document_class in SUPPORTED_DOCUMENT_TYPES do
    for document_page in document_class.pages do
        best_align_score <- 0
        best_align <- null_object

        for threshold <- 0 to MAX_THRESHOLD_VALUE do
            biggest_contour <- get_max_contour(find_contours(uploaded_image, threshold))
            if biggest_contour <= 0 then
                continue

            transform_matrix <- cv2.getPerspectiveTransform(biggest_contour)
            image_warped <- cv2.warpPerspective(uploaded_image, transform_matrix)
            match_score <- get_match_score(document_page.template, image_warped)

            if match_score > best_align_score then
                best_align_score <- match_score
                best_align <- image_warped

        if best_align_score > best_match_score then
            best_match_score <- best_align_score
            best_match <- document_class.init_object(document_page, best_align)

if best_match_score < MINIMUM_ALLOWED_SCORE then
    return null_object

document_ocr <- empty_map
```

```
for document_field in best_match.fields do
    image_field <- crop_image(best_match.image, document_field.coordinates)
    ocr_text <- document_field.filter(TESSERACT_API.getText(image_field))

    if ocr_text =/= null_object then
        document_ocr.add_map_entry(document_field.name, ocr_text)
        continue

    for enhancement_method in SUPPORTED_ENHANCEMENT_METHODS do
        enhanced_image <- enhancement_method(image_field)
        ocr_text <- document_field.filter(TESSERACT_API.getText(enhanced_image))
        if ocr_text =/= null_object then
            document_ocr.add_map_entry(document_field.name, ocr_text)
            break

document_ocr.add_map_entry("document_type", best_match.class_name)
return document_ocr
```

4. Results and Evaluation

Our system has undergone numerous changes and performance improvements from the initial implementation until today and we will separate them into iterations to better understand the gains and the results.

4.1. Iteration 1

On the first iteration, we tried a more raw approach to processing the images: once the image reaches the server, it will be processed, this step involves 2 objectives: finding a defined template (we currently have two template types: IDCard and DrivingLicense) that matches it and aligning it with the respective template. If the image has no match then a warning is displayed to the user and is asked to manually select the area he desires to extract text data from.

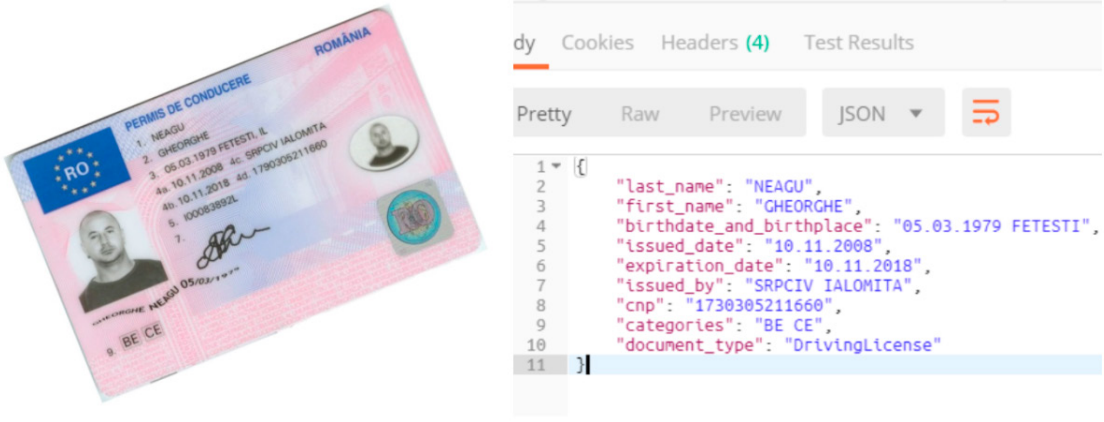

Fig. 2. Driving license data extracted by our system

Once the alignment is done, the background is removed from the image and its angle is corrected so that it almost matches the template. Later on, it will be resized if the dimensions don't match in such a way that the quality isn't affected negatively. After the image is processed and its template is known, the OCR tool will try to extract the data from it (see Fig. 2). Each template has a list of hardcoded coordinates that indicate the position of each relevant field. The component iterates over each set of coordinates and try to apply OCR to it. Each template has a list of regex for each field. The OCR is used and tries to extract the string for each field using its matching regex. Only if the resulting text is indiscernible, is the quality of the image is altered and tried again. A variety of mathematical methods will be

implemented to ensure a higher chance of success. If after N steps, the OCR still has no success then it will send a warning to the client that the extraction is not possible. For testing our solution properly we used a driving license that was rotated to show the precision of our system.

As we can see in Fig. 2, the data is very accurate but we had one major problem: it took 66.87 seconds to process all this data. That's a whole minute and it's not an acceptable time for an HTTP request. We started to analyze why this is happening and we found some major performance leaks with our RASE & SSIM scoring functions. To put that in perspective, we generated some graphs to better visualize time differences. We can observe that a ton of time is consumed by RASE and SSIM score functions, and to lower this time we brainstormed ideas to obtain a greater way of iteration over the image matrix in less time (see Fig. 3).

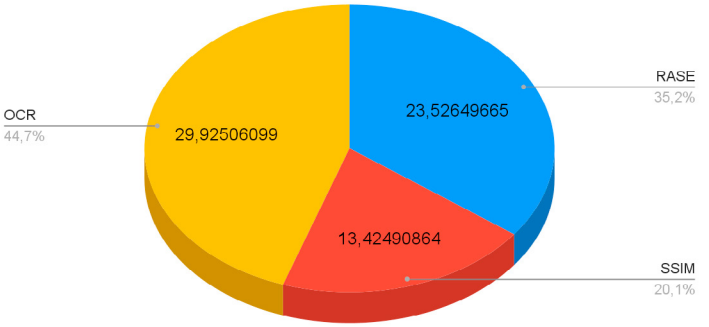

Fig. 3. First iteration processing time (66.87 seconds), viewing percentages

4.2. Iterations 2-3

The best idea that we had is to resize the image to a lower resolution [8] and check if the accuracy of the extracted text is still 100%. We first tried with 800×600 pixels and when we saw that the precision is still maximum, we tried further to reduce the size until we arrived at 200×100 pixels. From our tests, this is the sweet spot for the size reduction, given that if we try to reduce the resolution further, we drop some characters in OCR. When we finished the testing the results were amazing: we more than halved the total time and reduced the pre-processing time from an initial 36.96 seconds to 0.6 seconds (see Fig. 4).

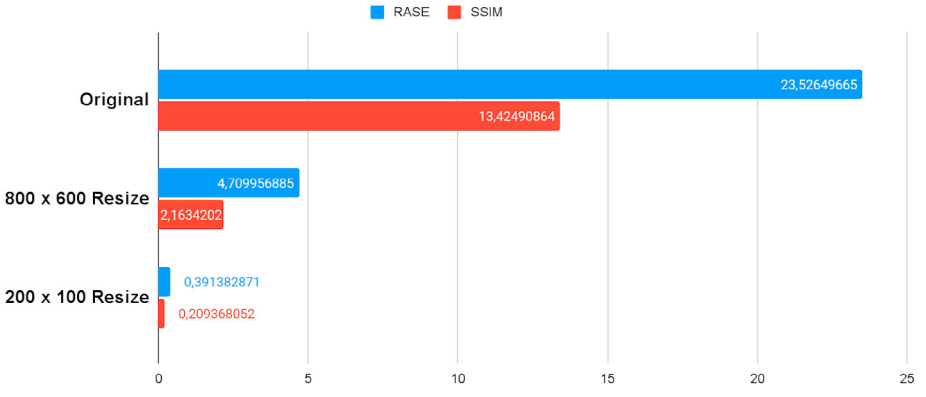

Fig. 4. Improvement on preprocessing using image resize

4.3. Iteration 4

After we solved the problem of preprocessing, the last thing that was very time heavy was the OCR and after many hours of research, we found one great article mentioned below that helped us understand that PyTesseract is just a wrapper on TesseractAPI [14] and this extra layer is very time-consuming so the first thing that we did to further improve our solution was to just transition to TesseractAPI and refactor our solution to take full advantage of the technology provided. The results were improved, in just one iteration we dropped OCR time from 29.69 to 6.47 seconds. These last two iterations were just magnificent and helped us reduce the total time from 66.87 seconds to just 7.11 seconds (see Fig. 5).

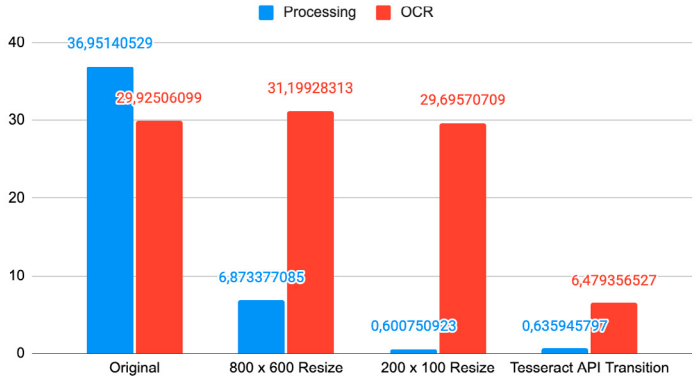

Fig. 5. Improvement on Preprocessing & OCR over four iterations

4.4. Iterations 5-7

Even if we improved our solution by over 900%, we still have more than 7 seconds for an HTTP call and as we saw in other solutions, 5 seconds would be a good result, but of course that the more we reduce this number, the better so we've further improved our solution by using a few more tricks (see Fig. 6).

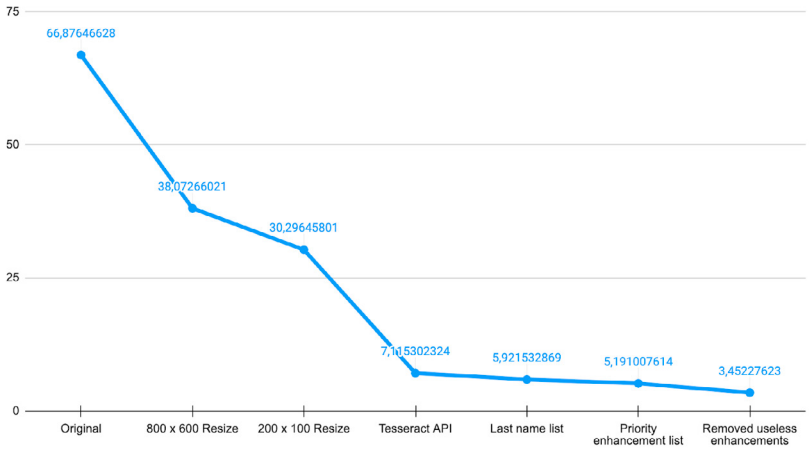

Fig. 6. Improvements on time through all the seven iterations

Firstly, we gathered a common name dictionary that we can use to help us with the OCR text matching over the fields, then instead of applying each of the enhancement methods over each of the fields, we empirically tested them

over a great set of data to prioritize them to obtain better results from the start to finish, not in a shuffled order and last but not least, if by applying a certain enhancement we reached our success rate threshold, then we remove any other enhancement from the list, to shortcut our way into results. By doing these three more things we preserved a 100% success rate on our initial testing document and further reduced the total time from 7.11 seconds to 3.45 seconds, this being a very good and pleasant result to see. This being said, after 7 iterations of improvements, we reduced the total time of the request from 66.87 seconds to just 3.45 seconds.

As we can see from Fig. 6, we've managed to make the application from a non-usable type of waiting time to very fast for its kind of business model. If you want to take a deeper look into our results, see Table 1 below.

Table 1. Data collected through all seven improvement iterations

		RASE	SSIM	OCR	TOTAL
Iteration 1	Original	23,526	13,425	29,925	66,876
Iteration 2	800 x 600 Resize	4,710	2,163	31,199	38,073
Iteration 3	200 x 100 Resize	0,391	**0,209**	29,696	30,296
Iteration 4	Tesseract API	**0,331**	0,305	6,479	7,115
Iteration 5	Last name list	0,359	0,260	5,303	5,922
Iteration 6	Priority enhancement list	0,370	0,253	4,569	5,191
Iteration 7	Removed useless enhancements	0,371	0,268	**2,813**	**3,452**

5. Future Work

Knowing the parts with which our system excels but also the less good parts of it, we intend to add improvements and micro-optimizations [3] in the future to make the recognition and extraction process even faster than it is at the moment. Also, a part where we have work to do is adding extra templates to have a more diversified portfolio of documents that can be identified by our product, thus obtaining more varied results and a larger user base. The last thing we propose is to build a web application to diversify the range of software interfaces, at this moment the only two methods of interaction with the system are through the Android application or directly calling the endpoints with HTTP requests. The information received on the Android app would not be saved on the user's device, it can be seen only until the app is closed and if the user needs the same information again, it is necessary to make a call to the server again. Adding another method of receiving the answer from the server through email would make the resulting information more accessible to the user, it would not be necessary to call the server every time he needs that information. For that, we have to make an email address for the server to send the answer and add functionality for creating an account and logging in in a secure way, so when a user wants to get the information from a document, he would be asked how he wants to receive the server's answer. This information should be sent to the server, besides the image and also this feature will enable us to export the information more easily in known formats (JSON, PDF, etc.).

6. Conclusions

Referring to the already existing solutions, seeing how a system for identifying and extracting data from official documents of this caliber is currently working and taking into account the existing time standards, we consider that the software built by us is very fast, easy to use, and accurate. Even though in the beginning the system was very efficient result-wise, the response times were too long to be considered a viable option, and the 7 iterations of improvements and refactoring managed to keep the accuracy and make the system over twenty times faster and place it over competing services like Sinosecu or Huawei Cloud Passport OCR. In addition to its accuracy and speed of response, the system works offline and offers complete transparency of processing which makes it a "must choose" by comparing it to competing systems. In conclusion, we consider that the results obtained are very good and that our application is easy to use, fast and efficient.

Acknowledgements

Data processing and analysis in this paper were supported by the Competitiveness Operational Programme Romania, under project SMIS 124759 - RaaS-IS (Research as a Service Iasi). Also, we would like to thank Alex Silistru for his contribution on developing this system.

References

[1] Mihai Baboi, Adrian Iftene, and Daniela Gîfu. Dynamic microservices to create scalable and fault tolerance architecture. *23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Procedia Computer Science, 4-6 September, Budapest, Hungary*, 159:1035–1044, 2019.

[2] Wojciech Bieniecki, Szymon Grabowski, and Wojciech Rozenberg. Image preprocessing for improving ocr accuracy. *International Conference on Perspective Technologies and Methods in MEMS Design*, pages 75–80, 2007.

[3] Yingxia Chen and Guixu Zhang. A pan-sharpening method based on evolutionary optimization and ihs transformation. *Mathematical Problems in Engineering*, pages 1–8, 2017.

[4] Todsanai Chumwatana and Waramporn Rattana-Umnuaychai. Using ocr framework and information extraction for thai documents digitization. *2021 9th International Electrical Engineering Congress (iEECON)*, pages 440–443, 2021.

[5] Ciprian-Gabriel Cușmuliuc, Ștefan Claudiu Susan, Bianca Demetra Chirica, and Adrian Iftene. News identification metric for classification prefiltering. *International Conference on INnovations in Intelligent SysTems and Applications (IEEE INISTA 2021), Kocaeli, Turkey, August 25-27*, pages 1–5, 2021.

[6] Markus Junker and Rainer Hoch. Evaluating ocr and non-ocr text representations for learning document classifiers. *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, 2:1060–1066, 1997.

[7] Taizo Kameshiro, Takashi Hirano, Yasuhiro Okada, and Fumio Yoda. A document image retrieval method tolerating recognition and segmentation errors of OCR using shape-feature and multiple candidates. *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99*, pages 681–684, 1999.

[8] Annika Kuhl, Tele Tan, and Svetha Venkatesh. Model-based character recognition in low resolution. *15th IEEE International Conference on Image Processing*, pages 1001–1004, 2008.

[9] Frank Lebourgeois. Robust multifont ocr system from gray level images. *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, 1:1–5, 1997.

[10] Sunghwa Lee and Jiwon Seo. Word error rate comparison between single and double radar solutions for silent speech recognition. *19th International Conference on Control, Automation and Systems (ICCAS)*, pages 1211–1214, 2019.

[11] Serena La Manna, Ana Maria Colia, and Alessandro Sperduti. Optical font recognition for multi-font ocr and document processing. *Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA*, 99:549–553, 1999.

[12] Prem Natarajan, Issam Bazzi, Zhidong Lu, John Makhoul, and Richard Scwhartz. Robust ocr of degraded documents. *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99*, pages 357–361, 1999.

[13] Tun-Wen Pai, Tieh-Ming Wu, Gan-How Chang, and Pei-Yih Ting. An intelligent chinese official document processing system. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 2:974–977, 1995.

[14] As Revathi and Nishi A. Modi. Comparative analysis of text extraction from color images using tesseract and opencv. *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 931–936, 2021.

[15] Foram P. Shah and Vibha Patel. A review on feature selection and feature extraction for text classification. *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 2264–2268, 2016.

[16] Harneet Singh and Anmol Sachan. A proposed approach for character recognition using document analysis with ocr. *Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 190–195, 2018.

[17] Harneet Singh and Anmol Sachan. A proposed approach for character recognition using document analysis with ocr. *Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 190–195, 2018.

[18] B. Srilakshmi and R. Sandeep. Shot boundary detection using structural similarity index. *Fifth International Conference on Advances in Computing and Communications (ICACC)*, pages 439–442, 2015.

[19] Yen-Min Su, Hsing-Wei Peng, Ko-Wei Huang, and Chu-Sing Yang. Image processing technology for text recognition. *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 1–5, 2019.

[20] Sahil Thakare, Ajay Kamble, Vishal Thengne, and U. R. Kamble. Document segmentation and language translation using tesseract-ocr. *IEEE 13th International Conference on Industrial and Information Systems (ICIIS)*, pages 148–151, 2018.

[21] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, apr 2004.

[22] Kei Watanabe, Shinji Takahashi, Yuhei Takagi, Masashi Yamada, Yoshito Mekada, Junichi Hasegawa, Takatoshi Naka, and Shinya Miyazaki. Detection of characters and their boundary from images of modern japanese official documents using fully cnn-based filter. *Nicograph International (NicoInt)*, pages 78–78, 2018.

[23] Jianxing Xu and Xing Wu. A system to localize and recognize texts in oriented ID card images. *IEEE International Conference on Progress in Informatics and Computing (PIC)*, 10:149–153, 2018.

[24] Cong Yao, Xiang Bai, and Wenyu Liu. A unified framework for multioriented text detection and recognition. *IEEE Transactions on Image Processing*, 23(11):4737–4749, November 2014.