COMBINING TESSERACT AND ASPRISE RESULTS TO IMPROVE OCR TEXT DETECTION ACCURACY

Remus PETRESCU [1*]
Sergiu MANOLACHE [2]
Costin-Anton BOIANGIU [3]
Giorgiana Violeta VLĂSCEANU [4]
Cristian AVATAVULUI [5]
Marcel PRODAN [6]
Ion BUCUR [7]

ABSTRACT

We live in a highly technologized era, where a great number of physical documents have started or must be digitized in order to make them accessible anywhere, for a great number of people. Optical Character Recognition (OCR) is one of the techniques which are widely used in order to recognize characters from specific images obtained after scanning. Different types of systems have been developed in order to perform Optical Character Recognition for various types of documents, but the task is not easy, as documents differ not in terms of content, but have also in formats, fonts, age or deterioration. After reviewing the existing systems, the paper at hand proposes one which uses two well-known OCR engines and a voting principle based on weights. There are also analyzed the results of our combined technique, as opposed to each individual approach of the two chosen engines.

KEYWORDS: *text recognition accuracy, OCR, Tesseract, Asprise, text detection*

[1*] corresponding author, Engineer, ”Politehnica” University of Bucharest, 060042 Bucharest, Romania, remus.petrescu28@gmail.com
[2] Engineer, ”Politehnica” University of Bucharest, 060042 Bucharest, Romania, xergiu90@yahoo.com
[3] Professor, PhD Eng., ”Politehnica” University of Bucharest, 060042 Bucharest, Romania, costin.boiangiu@cs.pub.ro
[4] Teaching assistant, PhD stud., Eng., ”Politehnica” University of Bucharest, 060042 Bucharest, Romania, giorgiana.vlasceanu@cs.pub.ro
[5] PhD stud., Eng., ”Politehnica” University of Bucharest, 060042 Bucharest, Romania, cristianavatavului@gmail.com
[6] Engineer, PhD Student, “Politehnica” University of Bucharest, 060042 Bucharest, Romania, marcoprod@gmail.com
[7] Associate Professor PhD Eng., ”Politehnica” University of Bucharest, 060042 Bucharest, Romania, ion.bucur@cs.pub.ro

INTRODUCTION

In the past years, a great number of written documents have been digitized, using scanners or different types of specific methods. The need for creating a system that can translate given data, such as pictures, to editable written documents, has appeared. The technology created for this task is Optical Character Recognition (OCR).

The problem with OCR engines is that a specific one may be good at recognizing only a specific type of scanned documents with certain characteristics - deterioration, paper quality, fonts and so on, and not always with 100% accuracy.

The focus of the presented research is set on how several types of OCR engines can be applied to a dataset of input scanned documents to yield the best result. In the next sections, we will analyze the potential result of each system and based on their performance on a specific document type, a voting algorithm will be employed in which the best engine has more weight in the overall decision process. We will start by giving details about the used systems and all possible alternatives. We will further present our system and analyze its workflow. Finally, we will present the results after the images were processed through our system.

USED SYSTEMS AND RELATED WORK

At the core of this paper is situated the OCR technology, which is the artificially reading process, in which image data from documents or natural scenes containing written messages is converted into text data [1].

Modern OCR engines are powerful because they provide the above-mentioned functionality without the need for developing the code further. The first OCR engines had to be trained on huge amounts of data to recognize characters or fonts and they were not always reliable because of the diversity and level of degradation that was present in specific datasets (skewed or blurred pictures, containing special characters, etc.). Some of the most widely used OCR engines, systems, models examples include:

- Ocropus [2] - or Ocropy, OCR engine based on LSTM;
- Ocrad [3] - The GNU OCR;
- SwiftOCR [4] - a fast and simple OCR library written in Swift;
- Attention-OCR [5] – a model for extraction of texts in real-world scenes;
- Tesseract [6] – mature OCR engine, including both NN and LSTM text recognition approaches;
- Asprise [7] - used as an OCR and barcode recognition SDK with high performance;
- Abby Finereader [8] – high-performance OCR and Layout detection engine

In this paper, we focused on the last two OCR engines described above, Tesseract and Asprise, and combined them to obtain better results. Each engine outputs a confidence level to measure if the detection process produces valid results.

Tesseract

Tesseract is a Google-developed OCR project, from 2006 [6]. It evolved a lot during the years, starting from a simple NN-based text reader, without any support for layout analysis, into a fully-featured system, which recognizes common layouts and offers both NN and LSTM recognition support. The later versions of Tesseract support different output formats, including hOCR with layout and formatting information [9], or may even be integrated with frontends such as Ocropus [10]. Even though initially designed to work for the English language and languages that read from left to right, Tesseract has been eventually trained to process different scripts, text orientations and reading orders [11].

Tesseract has no GUI and is run from the command-line interface, but several attempts to create one exist, such as OCRFeeder [12].

Despite having a lot of technical advancements, Tesseract has also several shortcomings, especially when we are taking into account input image page defects like:

- Invalid scanning resolution which results in less than 20 pixels font size;
- Artificially introduced skew, in the image acquisition process;
- Suboptimal image binarization due to changes in brightness across large areas, without significant edges;
- Extra-border surrounding the useful page data.

Asprise

The second system which we will describe is Asprise, a commercial OCR engine which has numerous abilities, including reading barcodes. It also possesses a number of features like:

- Solid text recognition, with quality that may be traded for speed;
- Multi-threading support and GPU acceleration, ensuring optimal use of the computing system resources;
- Multiple output formats.

SYSTEM WORKFLOW

After the analysis of Tesseract and Asprise, a new system is proposed, which uses both OCR engines and a voting mechanism based on weights to obtain the best output possible. Input files are processed through a series of steps, which can be observed in Figure 1. They are executed as follows:

- The system receives the input file and sends it to each OCR engine;
- Each OCR generates an output file with a certain confidence level - for each word in the case of Tesseract and for each row for Asprise (to make the data relevant we assigned the same confidence value to each word of the line), as seen in Figure 2);
- The system uses a reference test document in which all the correct words are manually inputted. The results from the system and manual set are compared in order to calculate the overall document correctness;
- Using the overall correctness of each output document, the system assigns a weight to each engine;
- The system combines the results based on the above-given weight.

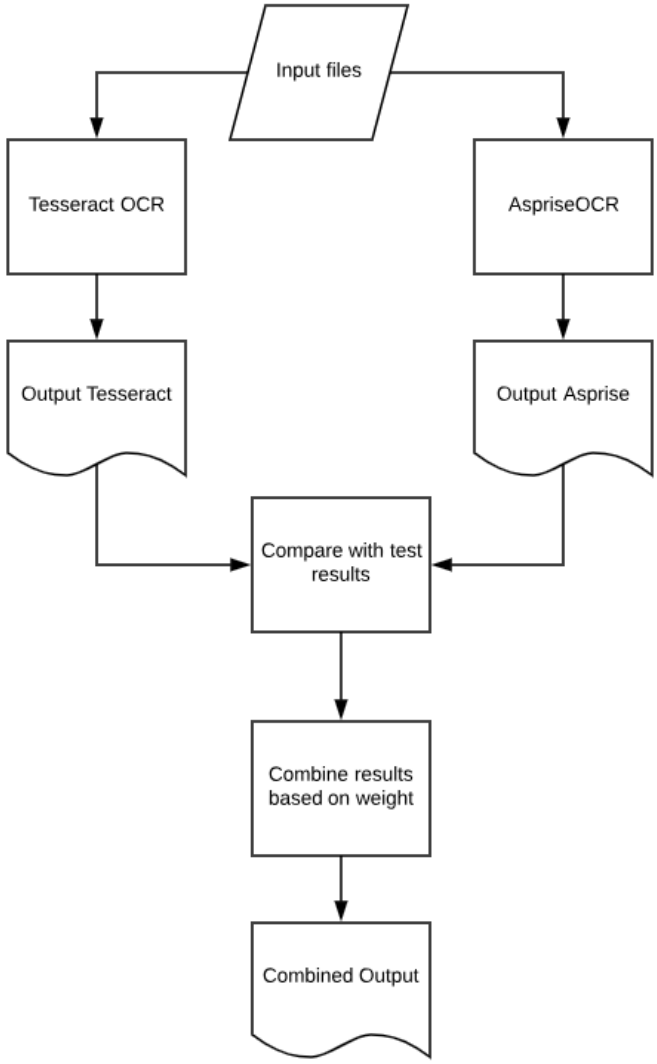

Figure 1. Steps of processing input files in the proposed system

[[('ENGLISH', 80), ('ROMAN.', 80)], [('(hgquue', 69), ('tandem', 69), ('abutére,', 69), ('Catilina,', 69), ('patientia', 69)], [("nofh'a?", 77), ('uamdiu', 77), ('nos', 77), ('etiam', 77), ('furor', 77), ('if'ce', 77), ('tuus', 77), ('eludct?', 77)], [('quem', 73), ('a', 73), ('fincm', 73), ('fefe', 73), ('efienata', 73), ('jaé'tabit', 73), ('audacia?', 73)], [('nihilnc', 79), ('tc', 79), ('nofiumum', 79), ('praefidium', 79), ('palatii,', 79), ('nihil', 79)], [('urbis', 82), ('vigiliae,', 82), ('nihil', 82), ('timor', 82), ('populi,', 82), ('nihil', 82), ('confou-', 82)], [('{us', 76), ('bonorum', 76), ('omnium,', 76), ('nihil', 76), ('hic', 76), ('munitifl'unus', 76)], [('ABCDEFGHIJKLMNOPQBSTVUW', 57)]]

[[('ENGLISH', 78), ('ROMAN.', 83)], [('@ufque', 52), ('tandem', 83), ('abutére,', 78), ('Catjlina,', 65), ('patientia', 76)], [("nofh'a?", 74), ('quamdiu', 79), ('nos', 83), ('ctiam', 78), ('furor', 76), ('if'ce', 70), ('tuus', 86), ('eludet?', 72)], [('quem', 84), ('ad', 88), ('finern', 87), ('fefe', 76), ('effremta', 76), ('jaéfabit', 64), ('audacia?', 81)], [('nihilne', 82), ('tr:', 74), ('noé'curnum', 66), ('praefidium', 76), ('palatii,', 82), ('nihil', 85)], [('urbis', 85), ('vigilia,', 62), ('nihil', 86), ('timor', 83), ('populi,', 82), ('nihil', 80), ('confou-', 73)], [('fus', 71), ('bonorum', 80), ('omnium,', 82), ('nihil', 76), ('hic', 85), ('munitiflimus', 80)], [('ABCDEFGHIJKLMNOPQRSTVUW', 53)]]

Figure 2. The output generated by Asprise (left) and generated by Tesseract (right)

RESULTS

After receiving the input file, each OCR has created its own visual interpretation of the text based on the confidence level.

Figure 3. The main scenario input test image

The input file is represented by Figure 3, whilst the output OCR texts color-bordered using the specific engine confidence in Figure 4.

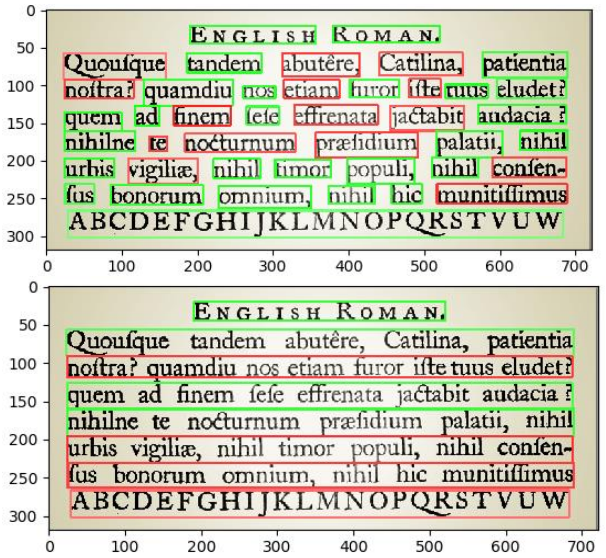

Figure 4. The visual output generated: Tesseract (top) and Asprise (bottom)

The following step was the analysis of the output files of each engine as seen in Figure 5 and their comparison to the test output in order to determinate and set the weight.

```
ENGLISH ROMAN.|(hgquue tandem abutére, Catilina, patientia|nofh'a? uamdiu nos etiam furor if'ce tuus eludct?|quem a fincm
fefe efienata jaé'tabit audacia?|nihilnc tc nofiumum praefidium palatii, nihil|urbis vigiliae, nihil timor populi, nihil
confou-|{us bonorum omnium, nihil hic munitifl'unus|ABCDEFGHIJKLMNOPQBSTVUW
ENGLISH ROMAN.|@ufque tandem abutére, Catjlina, patientia|nofh'a? quamdiu nos ctiam furor if'ce tuus eludet?|quem ad finern
fefe effremta jaéfabit audacia?|nihilne tr: noé'curnum praefidium palatii, nihil|urbis vigilia, nihil timor populi, nihil
confou-|fus bonorum omnium, nihil hic munitiflimus|ABCDEFGHIJKLMNOPQRSTVUW
```

Figure 5. The output generated by Asprise (top) and by Tesseract (bottom)

In the last step, we combined the documents based on their given weights and obtained the final output which can be seen in Figure 6.

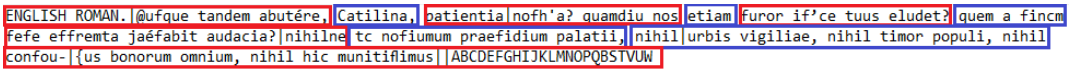

Figure 6. Results after combining the results (red Tesseract and blue Asprise)

A series of experiments were performed in order to assess the most appropriate weights. In figures 7 and 8 are identified some results based on the confidence level of each OCR engine. The intensity of the color gives the confidence level: green represents high confidence, red means low confidence.

Figure 7. The blurred scenario input test image

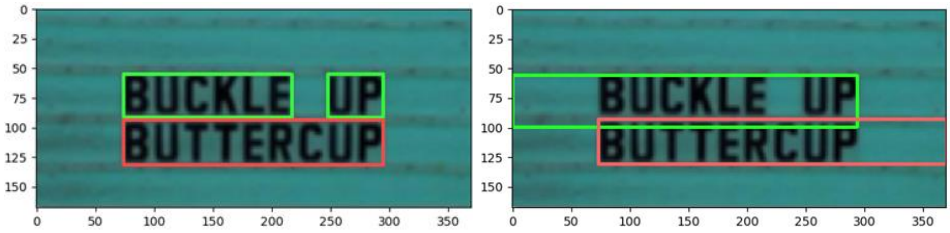

Figure 8. Output Tesseract (left) and Aspire (right)

After analyzing more situations, it is noticed that for any level of image blur, but especially for an intense one, Asprise tends to perform better than Tesseract. One good example is presented in Figure 9.

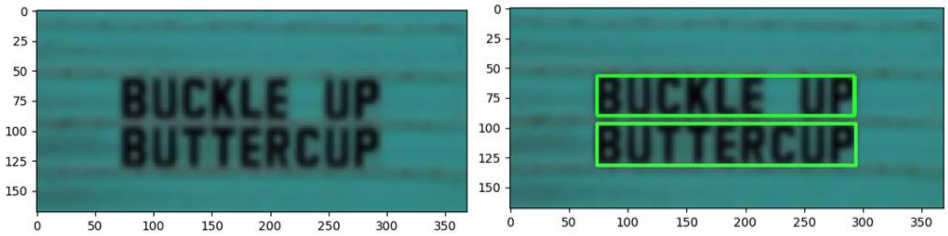

Figure 9. Output Tesseract (left) and Aspire (right)

CONCLUSIONS

When the results of several OCR engines were compared, it can be observed that each one of them has its own shortcomings when dealing with the degradation of the input files. In order to minimize the possible erroneous results given by the low-quality files which most engines will encounter, we decided to create a weight-based voting mechanism which analyzes both results and generate an output based on the confidences of the individual results.

The proposed approach improved the probability of creating a correct result by combining the individual engines' results into a single output, thus resulting in a higher detection accuracy than the individual output files.

ACKNOWLEDGEMENT

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI - UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0689 / „Lib2Life- Revitalizarea bibliotecilor si a patrimoniului cultural prin tehnologii avansate" / "Revitalizing Libraries and Cultural Heritage through Advanced Technologies", within PNCDI III.

REFERENCES

[1] Chaudhuri Arindam, Mandaviya Krupa, Badelia Pratixa, Ghosh Soumya K., Optical Character Recognition Systems, Springer International Publishing, 2017

[2] ocropy Wiki, URL: https://github.com/tmbdev/ocropy/wiki, Accessed on March 14, 2019

[3] Ocrad - The GNU OCR, URL: https://www.gnu.org/software/ocrad, Accessed on March 14, 2019

[4] Fast and simple OCR library written in Swift, URL: https://github.com/garnele007/SwiftOCR, Accessed on March 14, 2019

[5] Visual Attention based OCR, URL: https://github.com/da03/Attention-OCR, Accessed on March 14, 2019

[6] Tesseract OCR, https://github.com/tesseract-ocr/tesseract/wiki, Accessed on March 14, 2019

[7] Asprise OCR and Barcode Recognition, URL: https://asprise.com/royalty-free-library/ocr-api-for-java-csharp-vb.net.html, Accessed on March 14, 2019

[8] ABBYY FineReader Engine, URL: https://www.abbyy.com/en-us/ocr-sdk/, Accessed on March 14, 2019

[9] What output formats can Tesseract produce?, URL: https://github.com/tesseract-ocr/tesseract/wiki/FAQ#what-output-formats-can-tesseract-produce, Accessed on March 14, 2019

[10] Thomas Breuel, "Announcing the OCRopus Open Source OCR System", The official Google Code Blog entry, April 09, 2007, URL: http://googlecode.blogspot.com/2007/04/announcing-ocropus-open-source-ocr.html, Accessed on March 14, 2019

[11] OCR – Community Help Wiki, URL: https://help.ubuntu.com/community/OCR, Accessed on March 14, 2019

[12] OCRFeeder, URL: https://wiki.gnome.org/Apps/OCRFeeder, Accessed on March 14, 2019.