

# Automatic Correction of OCR Results Using Similarity Detection for Words and Fonts

Costin A. Boiangiu, Mihai Zaharescu, Oana Ferche, and Andrei Danescu

**Abstract**—This paper presents a novel approach for optimizing the results of a traditional OCR system, by implying a prior analysis of the input image document. The purpose is the interpretation of highly deteriorated, low resolution images. The idea behind this approach is to use text redundancy in order to estimate unclear areas. This is done at the image level, by replacing degraded areas with other regions likely to contain the same information. This replacement is done based on two systems: selection of the similar regions based on surrounding font statistics, constructing lists of deteriorated and clear words and choosing the best possible replacements.

**Keywords**—Font classification, K-Nearest Neighbors, Normalized cross correlation, OCR preprocessing.

## I. INTRODUCTION

**O**PTICAL Character Recognition (OCR) is a method for converting a raster digital image containing scanned text, into actual machine encoded text that the computer can understand and index [4]. To achieve this fascinating task of understanding the contents of a real world document, it must pass through multiple processing stages:

- 1) Conversion to raster digital form by scanning or photographing.
- 2) Preprocessing and cleaning up as much as possible in a phase that offers a grayscale raster image of the document. Here we can think of noise removal, specular removal or light uniformization, contrast stretching, decolorization [10], sharpening, deblurring [11] or super-resolution and multiple other filters that can generate a clear grayscale raster image, with very distinguishable text from the background [12].
- 3) Some OCR systems are able to work on these grayscale images, but the vast majority prefers a binary black and white image as input, not to mention that almost all geometry correction and page segmentation algorithms

work on binary images. Thus the next important step to be considered is a massive reduction in information: separation of the background and foreground through image binarization.

- 4) On the binarized image geometric correction algorithms can be applied, for unwarping distorted pages and deskewing rotated images in order to have horizontal text [13].
- 5) Page segmentation and interpretation algorithms follow on these binarized and corrected images. They are able to cut out images and graphs from text and also label some of the important text regions (titles, paragraphs, page numbers, etc.). Our proposed solution falls partially in this category, by selecting homogeneous text regions, and partially in the last two, by classifying words and correcting unrecognizable regions.
- 6) Only now these text fragments can be sent to the OCR for conversion to actual text through correlation, neural networks or feature similarities.
- 7) The last step is a correction of the low confidence words with a predefined dictionary. In our solution this step becomes less important, as many of the words will already be replaced by a correctly recognized version.

The interpretation of the text is made difficult by low contrast, inadequate brightness settings or lighting conditions, inconsistent use of font faces and sizes, small font sizes of below 6 points [14], typescript, handwritten or texts published prior to 1850 and for some languages even later, crooked lines of text, noise, etc. [12] An adequate image that is to be subject to OCR processing must meet certain requirements [1][5][6]. Once these requirements are not met, the process of conversion can fail at any of the stages mentioned earlier, the errors propagating in avalanche to the next stages.

Current recognition rates for good input documents, for commercial applications, are between 97% and 99%. But as soon as those conditions are not met, it can drop to as low as 0% for medium readable text.

The method we propose covers a wide range of the processing phases mentioned. As it is implemented now, it can be placed after the geometrical correction, as a less common segmentation, but also as postprocessing when replacing unrecognizable words, not from a dictionary but from the same source image, making this approach dictionary- independent. The proposed method in the future work section, that works also on the grayscale raster and geometry corrected image, can be placed a bit closer to the initial phases.

C. A. Boiangiu is with the Computer Science Department from “Politehnica” University of Bucharest, Romania (e-mail: costin.boiangiu@cs.pub.ro). The original research carried by C. A. Boiangiu was supported by Electronic Arts Romania S.R.L.

M. Zaharescu is with the Computer Science Department from “Politehnica” University of Bucharest, Romania (e-mail: mihai.zaharescu@cs.pub.ro).

O. Ferche is with the Computer Science Department from “Politehnica” University of Bucharest, Romania (e-mail: oana.ferche@cs.pub.ro).

A. Danescu is with the Computer Science Department from “Politehnica” University of Bucharest, Romania (e-mail: andrei.danescu@cti.pub.ro)

Our purpose is improving the OCR results of seriously damaged document scans, which cannot be corrected by filters and raster or geometric processing, by feeding the same word, positioned in different regions of the page, to the OCR for recognition. The patch containing the most visible word will be placed over all the other unrecognizable, but very similar regions. In this way, the damaged text regions won't be replaced with junk, and no post-processing needs to take place.

This method can also improve the speed of an OCR system, by running the recognition module on a single word rather than on every appearance of the same word, if the replacement is done at the text level. This, however, needs a fast method for selecting similar words.

As from the segmentation point of view, we want to firstly select regions that have similar characteristics, and not compare, as for example, italic with normal text. To obtain this we opted for a font similarity classifier at the character level: regions containing the same fonts being grouped for word findings. This is done by a K-Nearest neighbor [3] font classifier that selects the characters and assigns a font type to each letter. The classifier was previously trained with all the letters of the alphabet, for every font, with modifiers (bold, italic, underlined, strike-through etc.) [2].

In the end the cross correlation between segmented words and the initial uniform-font region can be applied to generate a list of similar regions that can be merged together.

In the following chapter the proposed solution will be presented and discussed upon. Then the prototype verification program results are presented and in the end a few directions for future research in optimizing this approach are offered.

## II. PROPOSED SOLUTION

The proposed solution consists of two major processing steps: the segmentation of the initial document in homogeneous regions on which further interpretation makes sense, and the actual word replacement technique for correcting damaged regions. Each of these phases has its sub-steps that are presented now.

### A. Phase 1: Segmentation by Font Classification

#### 1) Preprocessing

This stage deals with the preparation of the image for the classification process and the extraction of the letters to be classified. The operations are as follows:

- 1) Simple conversion to grayscale.
- 2) Noise filtering by a 3x3 Gaussian filter (other filters can be used here, but future steps can handle remaining artifacts).
- 3) Binarization, done through combining localized and global thresholds obtained through Otsu's method in order to have an adaptive threshold value.
- 4) Morphological filters, applied for reconnecting letter fragments resulting from the binarization. Filters include morphological erosion and dilation (closing). This operation helps in the case of thin fonts (e.g. Corsica) whose letters are "breakable" or poor contrast input image.

The result of this step is a binary image with mostly connected character fragments and some noise.

#### 2) Character Extraction

After this basic preprocessing, the image is ready for the contours' extraction. The method used here is the one described by Suzuki in [1], and the algorithm is based on border tracking, the implementation from OpenCV.

The binarized input image at this stage contains graphic elements, noise and other outliers. Our aim is not a good cleanup, but a strong cleanup that only keeps some of the existent characters. As text regions are mostly similar, only the correct recognition of a few characters is enough to classify a paragraph. To remove the outlines that remain after thresholding, the following filtering algorithm is applied:

- 1) We calculate the width and the height of each frame rectangle.
- 2) If the width fits the range  $[width/2, width \cdot 2]$  of a previously analyzed rectangle, a value associated to that rectangle is incremented. Same applies to height. Otherwise, the rectangle is added in the list of those analyzed associating a counter value.
- 3) Excessively large contours, as well as the ones excessively small as compared to the image size, are excluded from processing.
- 4) Having considered all the rectangles, select as average the one that got the most votes.
- 5) Eliminate all contours that do not fit in the average value of height and width (with a margin).

The reason for which we only considered the regions that are predominant on the page (average width and height) is that there is no point in running the similarity algorithm on paragraph regions that contain few words, as the probability of finding repeating words is very small.

This algorithm produces valid results for most cases, because there are many more characters in a scanned document than noise or letter-sized images.

In this step we also obtain the character dimensions which are useful in the second phase, when we cut out the words from the document.

Following this step, we obtain contours corresponding to the letters that need to be classified.

#### 3) Identifying Fonts

This step is performed using a neural network of type k-Nearest-Neighbor. In the field of machine learning, k-NN is a non-parametric method used for classification and regression. As starting parameters, it employs the closest k training examples in the characteristics' space.

In the case of classification, the k-NN output is represented by the class of the analyzed object [3]. An object is classified by a majority vote of its neighbors, and it has as objective its assignment to the class that is the most common among those of its k neighbors. If  $k = 1$ , the object is directly allocated to the class of its nearest neighbor.

In the training phase, vectors belonging to a multidimensional space of features, each with a class label, are

transmitted to the network. Training the algorithm consists of storing the values of the vectors and the class labels of the input values. In the classification stage,  $k$  and a vector of unlabeled features are offered in order to be allocated to a class.

The metric most often used in setting the closest neighbors is the Euclidean distance for continuous variables, and Hamming distance for discrete space.

Training the network is achieved by exploring every letter comprised in an entry set. The letters' images are resized at a predetermined dimension. Classifying each character in the text image follows the same method of image processing: each character is resized to the imposed dimensions, while preserving the relative aspect ratio. The  $k$  of the network in the classification stage is 1 and the input data are the pixel values.

The output of this step is assignation of every character to a font class.

#### 4) Text segmentation

Characters alone don't offer sufficient information; it is the global statistic that says if a paragraph is of a specific type. The characteristics of a paragraph are obtained by calculating the convex envelope of all letters belonging to that font.

The results of this method are disturbed by incorrect classifications. The envelope contains points located in different font blocks. To counteract this defect, various methods can be applied:

- 1) The analysis of the number of neighbors of each identified character, and its class reallocation based on the majority of the neighbors;
- 2) The segmentation of the convex envelope of a font based on K-Means algorithm with  $k > 1$ , while identifying a character from another font detected inside it;
- 3) The union of the surrounding rectangles of the close characters of same font.

The result of Phase 1 is a binary image, containing only regions having similar font sizes, having paragraphs of similar fonts encapsulated in a convex hull.

#### B. Phase Two: Finding Similar Words

This phase deals with the actual word corrections computed inside each of the similar regions separately.

##### 1) Template matching

The detection of similar words is done through Normalized Cross Correlation (NCC) which assigns a similarity score between all words.

Cross Correlation is a sliding product between two functions. It shows the similarity between the two waveforms at a shifted time. It is used to find a waveform in a larger signal. But because Cross Correlation uses the multiplication to estimate similarity, it is affected by the lightness or integral of the functions to compare, thus, lighter regions will seem more similar. In order to avoid this, normalization is applied by subtracting the mean and dividing by the scattering or standard deviation.

The NCC algorithm was implemented and used for finding a zone that would yield the best results in a template matching procedure. The created functionality receives as input two images: the first one is the target search domain, whereas the second is designated as the search pattern (template image), in our case, each word for which we want to find similar words in the homogeneous region it belongs to.

##### 2) Exact word finding

The results obtained for this step are presented in the following images.

Firstly, an image containing a lot of text is required, because it represents a valid search space to search for templates. Such an image is exemplified in Figure 1.

To overcome these complexity problems, an efficient method cross correlation coefficient is proposed by Lewis.<sup>5</sup> The main integral over the image function  $f(x, y)$  and the squared image  $f$ , and use these tables for efficient calculation of the expression normalized cross correlation coefficient is evaluated. Using this the denominator does no longer depend on the size of the template  $M_x, M_y$ . A brief description of this calculation is given in section 3.2

In section 3.2 the key idea, that allows a very efficient calculation. Thus, the normalized cross correlation coefficient (1) can be calculated with an order of magnitude less calculations than the standard method.

Fig. 1 The original image containing text (the search domain image). Source from [15]

Secondly, a template definition is needed, or more explicitly, a word to act as an image template to look for in the image domain (see Fig. 2).



Fig. 2 The word to search in text (the template image, the pattern to identify in the domain image)

The resulted image, after the search algorithm is applied, is presented in Figure 3; the best match position was visually marked by underlining.

To overcome these complexity problems, an efficient method cross correlation coefficient is proposed by Lewis.<sup>5</sup> The main integral over the image function  $f(x, y)$  and the squared image  $f$ , and use these tables for efficient calculation of the expression normalized cross correlation coefficient is evaluated. Using this the denominator does no longer depend on the size of the template  $M_x, M_y$ . A brief description of this calculation is given in section 3.2

In section 3.2 the key idea, that allows a very efficient calculation. Thus, the normalized cross correlation coefficient (1) can be calculated with an order of magnitude less calculations than the standard method.

Fig. 3 The resulted image after the search algorithm running and the underlining of the best correlation position. Source from [15]

It can be noted that a very good identification of the word position in text is obtained after the search operation is performed.

### 3) Similar words detection

Until now we offered the searching template manually. But the algorithm needs to select the templates automatically. This is done by selecting the words from the binarized image (a word segmentation algorithm [7]).

The approach used here is smudging on a limited distance, relative to the detected character dimension deduced in phase 1. Word letters will end up connected in the same block, while white-space between words will be larger than the smudge threshold. The individual words are saved in a "Word Database".

A first correlation phase takes place between the words in the database, in order to form the lists of similar words.

To make sure that the words that have been determined are unique, a correlation function is called for every possible word placement position in the input image. When a correlation greater than a threshold (in our experiments, 0.9 from 1) is found, the algorithm concludes that it has detected an instance

of the template word in the image and connects it to the word list.

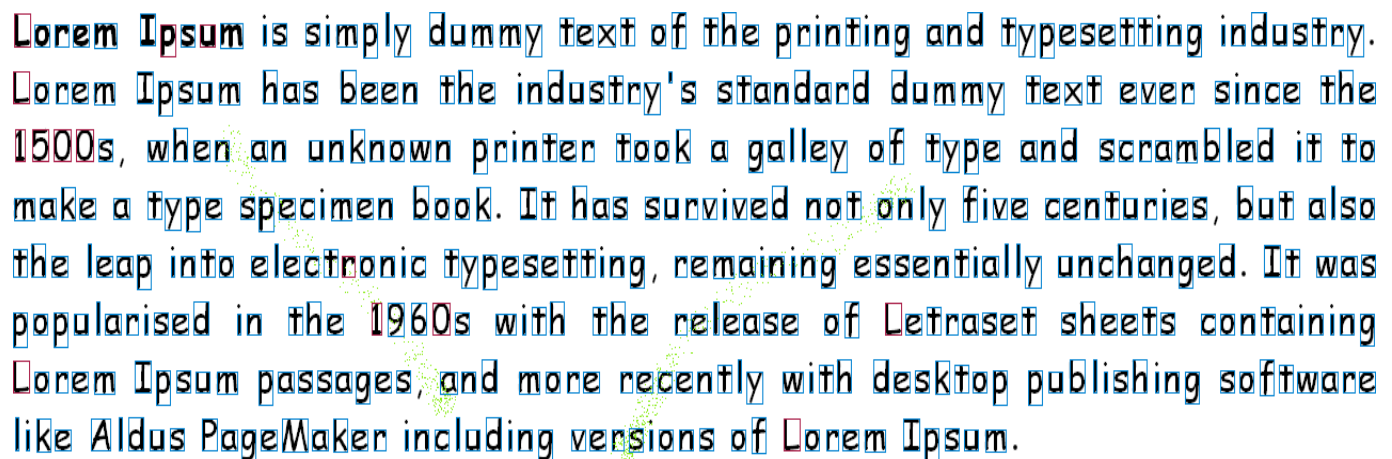
OCR can be applied now only on the word lists and selecting only the best recognized word in the list to replace all similar words.

## III. RESULTS

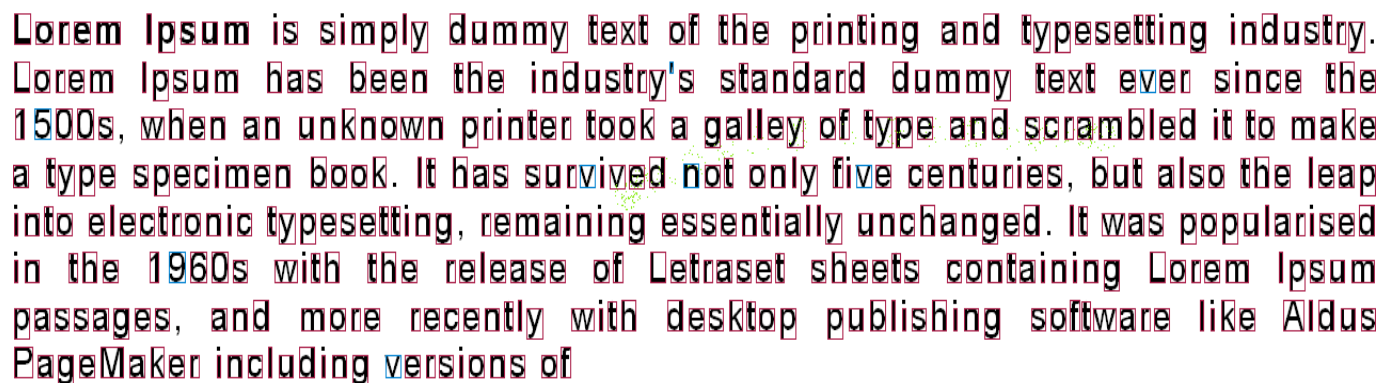
### A. Font-based Region Clustering

For testing the font based classification, we used Comic-Sans-MS and Arial fonts. A salt and pepper noise was applied to the input image and we also used images of different sizes. Average classification error was of 8%, due to the similarities between certain letters as well as the points associated with the characters  $i$  and  $j$ , detected as separate symbols.

Font accuracy is very good in this example (Fig. 4) and the two distinct paragraphs can be clearly differentiated.



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of

Fig. 4 Classified test image: Cyan = Comic-Sans, Purple = Arial.

### B. Word-based text clustering

Inside each of the uniform paragraphs individual words are found and precisely tagged in their corresponding random color. A visual example of the application result is shown in Fig. 5.

Before computing the correlation of a certain word, the

application chose a random color to associate to that individual word; in the moment a very good match position was found, the application replaced the word pixels with the aforementioned color.

The words having the same color belong to the same word-list that will be fed to the OCR.

acest text este un text de test un test un text  
 alt text de test un alt text de test acest text  
 acest test un alt acest text de test un text de  
 acest text este un text de test un test un text  
 alt text de test un alt text de test acest text  
 acest test un alt acest text de test un text de

Fig. 5 Input (top) and output (bottom) image resulted after the final step; every individual word is properly tagged using a unique random color

#### IV. COMBINING THE APPROACHES

The explanations for the important algorithms were given in the previous chapters. This chapter aims at integrating the modules in a system able to process the documents based on learning words from images.

The main problem with real world images, and especially old, deteriorated documents, is that the content is far from being clear to read, as the examples we used for describing the two main processing phases.

##### A. Input

In order to exemplify the low resolution processing phases, we start with a low resolution (100 dpi) photograph of an old deteriorated book. The input image size is 920x733 pixels and contains stains, warped text, and variable illumination and compression artifacts. It is not recommended to pass such an image to an OCR engine and still expecting reasonable results.

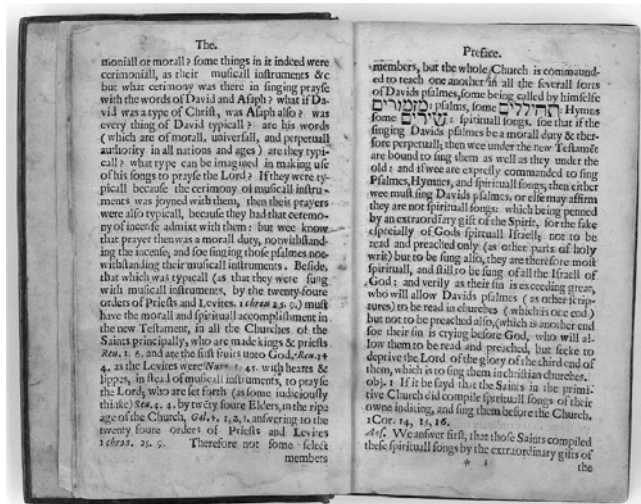


Fig. 6 Highly deteriorated, low resolution input image used for testing the system

##### B. Binarization

The images that we worked with so far were binary image, meaning the foreground was clearly separated from the background. Because binarization is not the purpose of this paper, in order to convert it to black and white, we applied a simple global Otsu binarization algorithm:

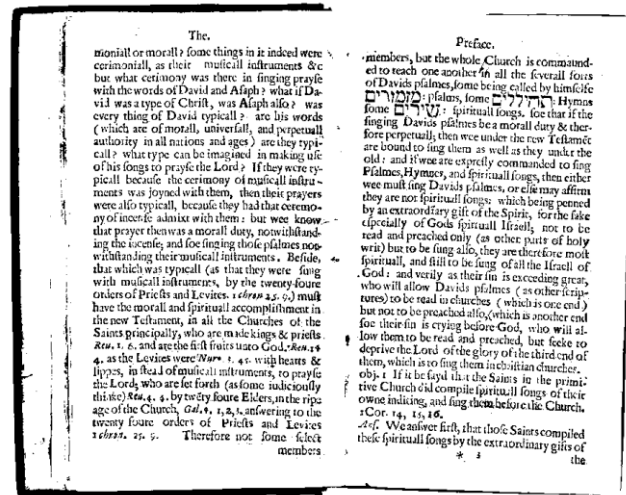


Fig. 7 Result after binarization

As it can be noticed, the input image contains more non-text objects than actual letters, and the letters themselves are either combined or broken and highly deteriorated.

This result can be strongly improved by using a more advanced binarization algorithm (here we can mention an adaptive algorithm that not only selects the threshold inside a local window, but also chooses the best size for each local window for each pixel, basing on statistics, in order to contain equal amounts of foreground and background. Though it may seem very expensive, we tested this approach, optimized on the GPU, and it offers a result for a 63 megapixel image in one minute). This approach, not containing the optimization, is detailed in [17].

##### C. Similar Font Region Detection

The first phase in the mentioned pipeline is selecting regions of text having similar fonts. However, the neural network algorithm used by us would have trouble clustering textual elements containing as much non-textual data as in this input image. Thus a first pre-filtering is needed. The proposed method preserves only the most common size elements from the page.

For this example 2480 elements were deduced. Everything exceeding a font size of 36 and below 5 was removed. Using the remaining elements the most common font size was calculated, as the mean of the objects' sizes, and that font size was the only one kept. Because some of the letters were split, this process resulted in the elimination of useful text information.

However, this problem can be solved by inserting a text filtering algorithm in the pipeline. This includes grouping together fragments that are likely to be part of the same letter, basing on:

- 1) inside filter (combines elements that are included in the bounding box of a larger element)
- 2) merge filter (combined elements belonging to the same line of text and are one on top of one another (ex. the dot from



the letter “i”)

3) width filter (large width elements most likely contain merged letters that need to be split at the thinnest bridge point that keeps them equal in size)

The process is described in more detail in [16]. Alternate processing may be performed using a similar approach described in [20].

Even without applying these steps, the result is still very good for our demonstration purpose.

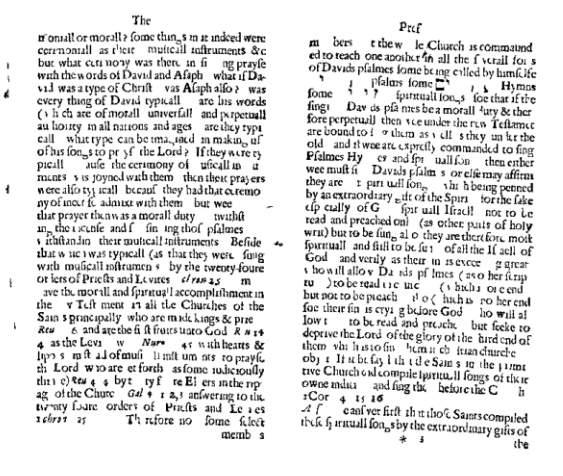


Fig. 8 Results after object filtering

The page now contains only elements having similar size, thus the font classifier now has a much simpler task. It can be noticed that the Arabic symbols at the top of the right page are now also gone.

This image is now the input for the font detection algorithm. Because of the very small resolution, the system isn't able to make a finer classification than italic and non-italic text. Because of the noise levels in the shape of the text, the class outputs of the neural network are limited to a very small number.

In order to automatize the process of choosing between a full font search and only basic features search, the output obtained when using the full training set is used by noticing the very high variety of fonts obtained. In this situation, an italic,

bold, simple and non-text separation is enough. In our situation, most of the elements are textual and non-italic, and the entire page is sent to the next processing step - similar word detection. If an entire paragraph contained bold or italic elements, the page would have been split in multiple regions needing to be processed separately. This process is accomplished by applying closing operation on the words and selecting the paragraphs. The paragraph's font is considered the font of the most numerous similar words.

D. Similar Word Detection

For demonstration purpose, a very simple approach for word detection was used: Smudging the image with a distance dependent on the letter size (one pixel less than half font size was experimentally observed to work good in order to keep words connected and not merge adjacent words, either from the same lines of from neighboring lines). The smudging is done only in the text direction (which can be automatically detected via projection profiling methods).

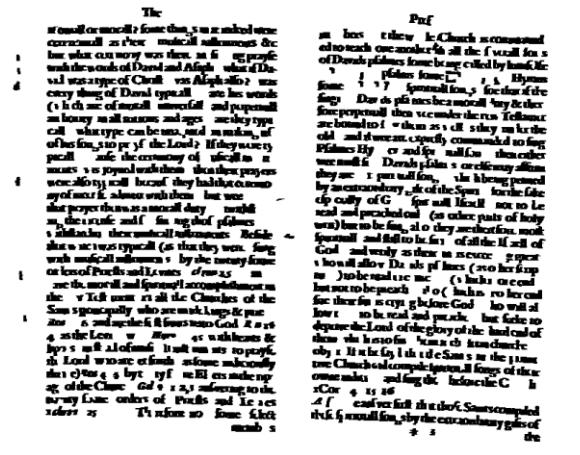


Fig. 9 Results for word segmentation

Now the connected elements are encapsulated in bounding boxes and saved as a list of words. The list of words is generated from the original, unprocessed image, by cropping the bounding boxes of the connected components.

psalm answering are therefore another in instruments perpetuall the and and accl the up  
 ny of before the be preach songs by imagiaed instruments the uall piri &c the onl  
 that t ceremoniall perpetuall the Spiri Churches preached her inst Gal by yf incer  
 commanded cerimoniall perpetuall the words spirituall psalmes exprefly the Da the but end vites the  
 spiritual command the words spirituall psalmes exprefly the Da the but end vites the  
 spiritual kings & uniuersall of Davids ceremony typicall the by pf: of the vhi tive  
 deprive typicall making if wee are psalmes instrumen for the the 2,3 one was  
 Hymns had that or morall preache the Lord joyned was fin of of of use  
 musicall the Church himselfe churche hority ith the ids thi not Ren  
 penned Church Pfalmes prayfe prayfe things mes wee ler be be ag  
 psalms Davids songs as some of musi being ng are be of be be  
 being musicall musicall estamer before songs all El of all can fin  
 uall fon usicall prayers twithst Churc admixt be not 16 are vee new  
 thing picall becaufe are the irituall indeed If of ad ell ren ure  
 thing picall becaufe are the irituall indeed If of ad ell ren ure  
 morall affirm morall orders Priests twenty God with their God them fore  
 songs glory is to fin verily Christ for the type with Sain hird them what  
 prayer Beside Israell hem it refore morall spir obj gift fake with their ive nts of if fu of  
 holy sung bound cially select David God type then then with read are are um ver are  
 memb morall singi of his of the ceremo read high Sain ause that lmes are ill 25 ch th no  
 sung if Da- David words thenw Pref be a also ments owne bers also al If 14 are fi to  
 icense scrip e)Ren inditi some either ther ages that low itian The to to le ds T ty  
 hearts oniall seeke Lord forth duty esp Hy then unto that excee ty ue ch al m to  
 other they song writ) every they first and icall and allo cry1 to or no f. T  
 foure their those called some some pfa and old still and and tu A m au to to  
 Lord hiehi in the they under some but and ment rmi uall will to to ft to in as  
 fruits chran verall duty they sing prie were were vid and but in et in in es as  
 they rCor them some they nony and but foe end by say m in as ft se se  
 byt these there what what must were call and and and and and and and and  
 them parts their their eanf teach end the Th fe the foe ( fi h fi w in  
 nations most great made their thok the the but the her the R 4 B 5 fi h

Fig. 10 Words extracted from the input document

Most of the words are recovered correctly, the problems noticed here being a few connected words and more split words.

Considering that we used only basic methods for obtaining necessary input data, the result is impressive. It can be however greatly improved by adding the aforementioned processing steps in the pipeline.

However, the problem that we encountered on the real world data is that even for a full text page, in a domain that words are likely to repeat, they do not repeat enough. A list of some of the most popular similar words found shows that only a small fraction of the words from the page actually repeat and those are mostly connection words:

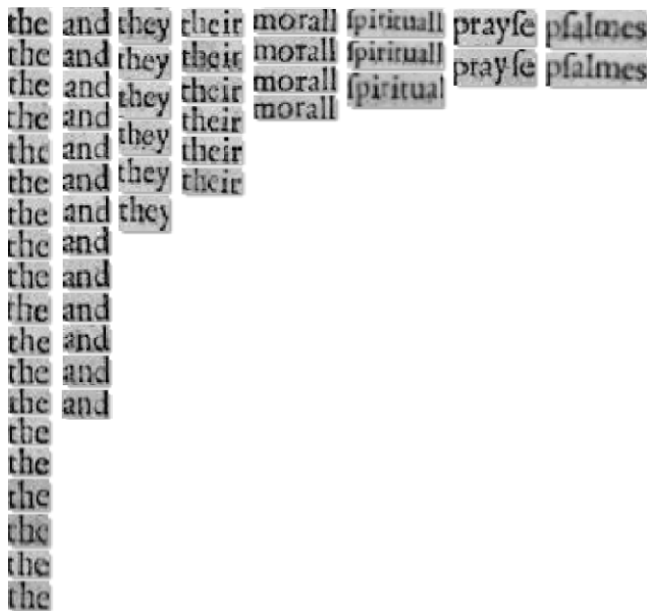


Fig. 11 A list of some similar words

By optimizing the system, the list can be grown with some words that were not identified correctly because of unification or segmentation. Examples include:

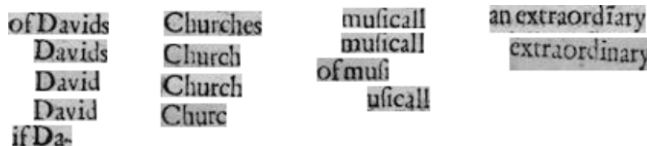


Fig. 12 Wrongly classified words

However, repeating words account only for a small number of the entire text, considering only one page of scanned document. And usually those words are the ones the OCR post processor is able to correct without the need of a separate system. This draws us to the conclusion that this system becomes useful only when converting multiple pages (like a book), in order to increase the chances of repeating words.

However, this brings a new problem: how will the word classification be made, because correlating each word in the list with each page from the book is too expensive.

In order to cope with this problem, some pre optimizations must be taken:

The correlations are done only between segmented words (so each word will be correlated with all the other words, after a word list is generated from all the files).

Only similar sized words are worth correlating. A difference of more than one letter size in word length forces those words to fall in different bins. So the word list is firstly sorted by width, this being the first classification. Words of similar width being considered the same, until a later phase, when only words from the same bin are measured for differences.

Because some words can be very similar from the correlation point of view, for example "and" and "end", it is better to sort the words based on similarity, and as soon as the

similarity score has a discontinuity, or an abrupt drop, the words must be considered different, even if we risk to create two identical bins. The idea is that it is better to have repetitive bins than to risk to merge two bins, the first introducing only a small performance issue, but the second a wrong result. This is achieved not by choosing a similarity threshold, but by inspecting the similarity variation between the similar words.

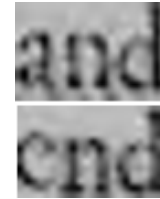


Fig. 13 Two different but similar words

Because the bounding box system centers almost perfectly the words, the correlation can be entirely skipped, a simple difference being enough for measuring similarity. In the future we will try to imagine a hash function, resistant to small variations, so the classification will be done by comparing the hashes after the widths. This will decrease the processing time a lot, as there will be only number comparisons instead of image comparisons on the long original lists, and image comparisons will be used only inside very small bins.

## V. CONCLUSIONS

The purpose behind this system was to be integrated with an OCR engine for improving results on very deteriorated documents. The correlation system builds a list of similar words, the OCR tries to recognize at least one word from the vector and replaces all the occurrences with the recognized word. In this way, the damaged words won't produce junk on the output and unrecognizable words can be replaced with the most likely variant, without the need of using a post processing step based upon dictionaries.

Real images were used for testing and the results show that this is a valuable direction for further research. Based on the test results, optimizations that can make the system more accurate and faster are proposed.

## VI. FUTURE WORK

The current version of the code uses normalized cross correlation for estimating the similarity between words. The reasoning behind choosing this slow approach is that the words are not placed identically inside their bounded boxes, and the y also may be warped or rotated. Noise is also a factor. However, upon testing, the differences were so small that a simple subtraction can be used. If however higher accuracy is needed, a fast variant is available: using the Fast-NCC [7]. The fast algorithm uses a formula for standard deviation based on the sums and squared sums in the image. These can be computed only once in  $O(N)$ , by generating the run-lengths of the elements and squared elements.

The brute force normalized cross correlation formula is:



$$\frac{1}{n} \sum_{x,y} \frac{(f(x,y) - \bar{f})(t(x,y) - \bar{t})}{\sigma_f \sigma_t}$$

where  $n$  is the number of sample points,  $\sigma_a$  is the standard deviation of  $a$  and  $\bar{a}$  is the meaning of  $a$ .

This works in  $O(N^2)$  because the mean and standard deviation are also computed in  $O(N)$ .

The mean can easily be optimally calculated by pre-generating a run-length image, in which each pixel,  $p(x,y)$ , has the value equal to the sums of all the pixels contained in the rectangle  $(p(0,0), p(x,y))$ . Then the mean is just computed by interrogating four values, in  $O(1)$ .

A bit more complicated is for the standard deviation, but a rewrite of the equation shows this is the same problem:

$$\sigma^2 = \frac{1}{n-1} \left( \sum_{x,y} f(x,y)^2 - \frac{\left( \sum_{x,y} f(x,y) \right)^2}{n} \right)$$

Then the statistics can be interrogated for every pixel in  $O(1)$ . If the words are very deteriorated, the FNCC algorithm positions the words one on top of the other and a merging between them, at pixel level, will probably yield an image the OCR will understand better. Having multiple low resolution images, super-resolution and denoising can be applied. If the merging is done for the binarized images, a voting approach, at pixel level, will generate a single image clearer than each individual one.

One of the most important aspects for the future would be to remove the comparison phase altogether, at least at root of the classification tree. This can be achieved by constructing a hash from each word image. The hash must be resilient to small changes, small variations in the image generating a small difference in the output number. An initial idea would be to use the sum of the 2D gradient at different scales, the most important weight being offered by the smallest resolution. Using this approach, firstly the words will be classified based on length, then only for the same length words, a classification based on hash will be applied and inside the hash FNCC or differences at the image level can be applied.

Other aspects include extra processing phases included in the pipeline for optimizing the result, or the utilization of more performant algorithms. These include smart binarization, rotation detection, connected letter splitting and split letter grouping. More advanced techniques like noise removal, light uniformization, super resolution and deblurring on the input image are also useful.

## REFERENCES

- [1] S. Suzuki, K. Abe, Topological structural analysis of digitized binary images by border following, *Computer Vision, Graphics, and Image Processing*, Volume 30, Issue 1, April 1985, pp. 32-46, [http://dx.doi.org/10.1016/0734-189X\(85\)90016-7](http://dx.doi.org/10.1016/0734-189X(85)90016-7)
- [2] M. Pietikainen, O. Okun, Edge-based method for text detection from complex document images, *Document Analysis and Recognition*, 2001. Proceedings. Sixth International Conference on , vol., no., pp. 286-291, 2001, doi: 10.1109/ICDAR.2001.953800
- [3] M. Abaynarh, H. Elfadili and L. Zenkour, Handwritten Characters Classification Using Neural Networks, *International Journal of Modern Engineering Research (IJMER)*, Vol.2, Issue.5, Sep-Oct. 2012, pp. 3572-3577
- [4] J. Gilavata, R. Ewerth, B. Freisleben, A Robust algorithm for Text Detection in images, *Proceedings of the 3rd international symposium on Image and Signal Processing and Analysis*, 2003, Vol.2, 611 - 616
- [5] S. A. Angadi and M. M Kodabagi, Text region extraction from low resolution natural scene images using texture features, *2<sup>nd</sup> International Advance Computing Conference*, IEEE, 2010, pp. 121 - 128
- [6] S. M. Hanif and L. Prevost, Texture based Text Detection in Natural Scene Image: A help to blind and visually impaired persons, *Conference & Workshop on Assistive Technologies for People with Vision & Hearing Impairments Assistive Technology for All Ages CVHI 2007*, M.A. Hersh (Ed.).
- [7] J. P. Lewis, Fast Normalized Cross-Correlation, *Vision Interface*, pp. 120-123, 1995.
- [8] D. M. Tsai and C. T. Lin, Fast normalized cross-correlation for defect detection, *Pattern Recognition Letters*, vol. 24, pp. 2625-2631, 2003.
- [9] D. M. Tsai, C. T. Lin, and J. F. Chen, The evaluation of normalized cross-correlations for defect detection, *Pattern Recognition Letters*, vol. 24, pp. 2525-2535, 2003.
- [10] A. Tigora, C. A. Boiangiu, Image Color Reduction Using Iterative Refinement, *International Journal of Mathematical Models and Methods in Applied Sciences*, NAUN, Volume 8, 2014, pp. 203-207
- [11] M. Zaharescu, C. A. Boiangiu, An Investigation of Image Deblurring Techniques, *International Journal of Mathematical Models and Methods in Applied Sciences*, NAUN, Volume 8, 2014, pp. 75-83
- [12] J. M. Booth, J. Gelb, Optimizing OCR Accuracy on Older Documents, A Study of Scan Mode, File Enhancement, and Software Products, Revised June 2006 (v.2)
- [13] W. Bieniecki, S. Grabowski, W. Rozenberg, Image Preprocessing for Improving OCR Accuracy, *International Conference on Perspective Technologies and Methods in MEMS Design - MEMSTECH*, 2007
- [14] C. Jacobs, P. Y. Simard, P. Viola, J. Rinker, Text recognition of low-resolution document images, *Proc. ICDAR*, pp. 695--699, 2005
- [15] K. Briechle, U. D. Hanebeck, Template matching using fast normalized cross correlation. *Proc. SPIE 4387*, Optical Pattern Recognition XII, 95 (March 20, 2001); doi:10.1117/12.421129
- [16] C. A. Boiangiu, A. Topliceanu, I. Bucur, *Efficient Solutions for OCR Text Remote Correction in Content Conversion Systems*, Journal of Control Engineering and Applied Informatics, Vol.15, No.1 pp. 22-32, 2013
- [17] C. A. Boiangiu, A. Olteanu, A. V. Stefanescu, D. Rosner, A. I. Egner (2010). „Local Thresholding Image Binarization using Variable-Window Standard Deviation Response” (2010), Annals of DAAAM for 2010, Proceedings of the 21st International DAAAM Symposium, 20-23 October 2010, Zadar, Croatia, pp. 133-134
- [18] C. A. Boiangiu, M. Zaharescu, O. Ferche, A. Danescu, “Improving OCR by Detecting Similar Words in Similar Fonts”, Proceedings of the 6th International Conference on Applied Informatics and Computing Theory (AICT '15), Salerno, Italy, June 27-29, 2015, WSEAS Press, pp. 74-80.
- [19] R. Bernhaupt, S. Schönert, “Using artificial neural networks as an image segmentation module of an OCR-system: A Preliminary Study”, ID no. 675, WSEAS NNA-FSFS-EC 2001, February 11-15, 2001, Puerto De La Cruz, Tenerife, Spain.
- [20] A. Momin, Sharad, S. Sanyal, “Novel Approach for Segmenting Fused / Merged Characters During Character Segmentation”, WSEAS International Conference on Information Technology and Computer Networks (ITCN '12), Vienna; 2012, pp. 344-348.