Recognition of Concordances for Indexing in Digital Libraries

Simone Marinai(✉), Samuele Capobianco, Zahra Ziran, Andrea Giuntini, and Pierluigi Mansueto

University of Florence, Firenze, Italy
{simone.marinai,samuele.capobianco,zahra.ziran}@unifi.it,
{andrea.giuntini,pierluigi.mansueto}@stud.unifi.it

Abstract. We describe a system for the automatic transcription of books with concordances. Even if the recognition of printed text with OCR tools is nearly solved for high quality documents, the recognition of structured text, where dictionaries and other linguistic tools can be of little help, is still a difficult task. In this work, we propose to use several techniques for correcting the imperfect text recognized by the OCR software by taking into account both physical features of the documents and the redundancy of information implicit in concordances.

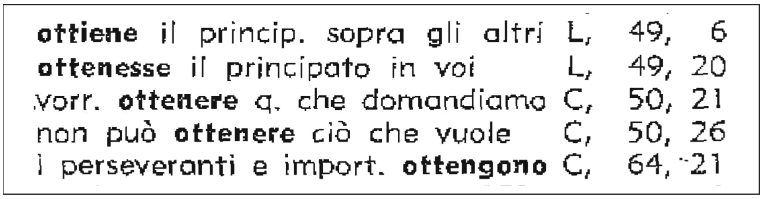
1 Introduction

One of the primary aims of digital libraries is to collect and organize information which can be accessed all over the world. Clearly digital information requires less space than paper based information reducing the library costs. Another obvious advantage is the reduction of the access time to the information. Digitization is also a way to preserve historical documents from the damage and wearing effect of time. Having digitized documents it is clearly possible to use automatic techniques to analyze and extract structured information.

When a word list is an index to a limited body of writing, with references to each passage, it is called a concordance [9]. In general, constructing concordances has been very difficult and time consuming. The term concordance originally applied just for indexing the Bible. Its root (i.e., concord) means unity and the term seemingly arose out of the school of thought that the unity of Bible should be reflected in consistency between the old and new Testaments and could be demonstrated by concordances. The 1960s and 1970s saw another uprising of concordance made possible by the ease with which indexes could be constructed with the use of computers. Concordances then came into being for all types of literature, from Charlotte Bronte to Dylan Thomas. More recently, it has become possible to accomplish this task effectively by the use of Optical Character Recognition [12].

In this work we deal with one digitized volume which is stretched from the *Concordances of the writings of S. Antonio M. Zaccaria* [1], which consists of the verbal concordances that refer to the holy volumes *Letters*, *Sermons* and

© Springer Nature Switzerland AG 2020
M. Ceci et al. (Eds.): IRCDL 2020, CCIS 1177, pp. 135–147, 2020.
https://doi.org/10.1007/978-3-030-39905-4_14



ottiene il princip. sopra gli altri	L,	49,	6
ottenesse il principato in voi	L,	49,	20
vorr. **ottenere** q. che domandiamo	C,	50,	21
non può **ottenere** ciò che vuole	C,	50,	26
i perseveranti e import. **ottengono**	C,	64,	21

Fig. 1. Example of concordances in *Concordances of the writings of S. Antonio M. Zaccaria.*

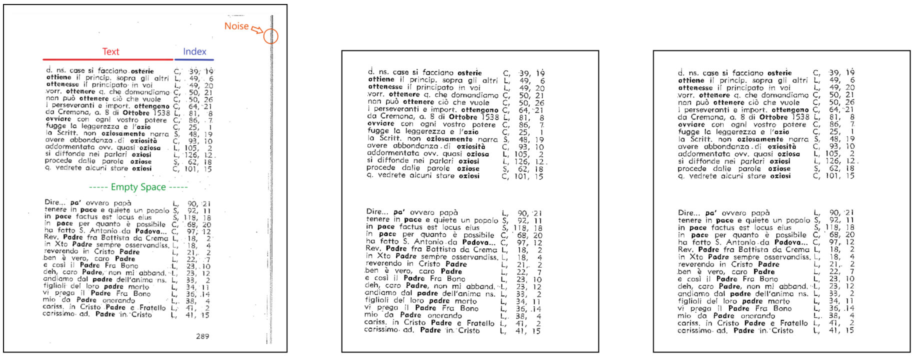
Constitutions. The document is 442 scanned pages long and each one has a regular structure containing a given number of entries of the concordances. To better illustrate the concordances in general and in particular for this volume we describe in the following the example shown in Fig. 1 that is extracted from one page. As we can notice, the five lines correspond to five occurrences of the verb "**ottenere**" or different verb tenses. For each occurrence (printed in bold), the text line in the left shows its context (the words around the given occurrence). On the right part we can find the position of the occurrence; this part can be considered as an inverted index for this information: the first column corresponds to the book (L: *Letters*, S: *Sermons*, C: *Constitutions*), the first number to the page, and the last to the textline in the page.

By closely inspecting the example, we can notice some features of the problem that can help understand the motivations for designing the tool described in this work and some of its main peculiarities. First of all the scanning resolution is rather low and therefore a state-of-the-art Optical Character Recognition (OCR) tool leads to some errors. These errors are particularly difficult to detect and to correct especially in the case of numbers in the right part of the page. However, one important peculiarity of concordances (printing the context of the indexed word) can help us since in most cases the same textline is indexed several times and we can exploit this feature in the recognition as detailed in Sect. 2.

1.1 Previous Work

Page segmentation is performed on document images to extract homogeneous components from one page image. Physical layout analysis is a combination of page segmentation and classification. Important applications of layout analysis are text-line or word segmentation from printed documents. Several techniques have been presented to address text line segmentation on historical handwritten documents [2,5], however, one widely adopted solution to detect textlines in printed documents relies on projection profiles [3]. State of the art text recognition is on the other hand achieved with optical character recognition (OCR) tools like Tesseract [10] which includes line finding, features/classification methods, and an adaptive classifier.

Automatic table of content (ToC) recognition is another related task in document image analysis research. One example is [7] where the authors propose a



d. ns. case si facciano **osterie**	C,	39,	19
ottiene il princip. sopra gli altri	L,	49,	6
ottenesse il principato in voi	L,	49,	20
vorr. **ottenere** q. che domandiamo	C,	50,	21
non può **ottenere** ciò che vuole	C,	50,	26
i perseveranti e import. **ottengono**	C,	64,	21
da Cremona, a. 8 di **Ottobre** 1538	L,	81,	8
ovviare con ogni vostro potere	C,	86,	7
fugge la leggerezza e l'**ozio**	C,	25,	1
la Scritt. non **oziosamente** narra	S,	48,	19
avere abbondanza di **oziosità**	C,	93,	10
addormentata ovv. quasi **oziosa**	L,	105,	2
si diffonde nei parlari **oziosi**	L,	126,	12
procede dalle parole **oziose**	S,	62,	18
q. vedrete alcuni stare **oziosi**	C,	101,	15
Dire... **pa'** ovvero papà	L,	90,	21
tenere in **pace** e quiete un popolo	S,	92,	11
in pace factus est locus eius	S,	118,	18
in **pace** per quanto è possibile	C,	68,	20
ha fatto S. Antonio da **Padova...**	C,	97,	12
Rev. **Padre** fra Battista da Crema	L,	18,	2
in Xto **Padre** sempre osservandiss.	L,	18,	4
reverendo in Cristo **Padre**	L,	21,	2
ben è vero, caro **Padre**	L,	22,	7
e così il **Padre** Fra Bono	L,	23,	10
deh, caro **Padre**, non mi abband.	L,	23,	12
andiamo dal **padre** dell'anima ns.	L,	33,	2
figlioli del loro **padre** morto	L,	34,	11
vi prega il **Padre** Fra Bono	L,	36,	14
mio da **Padre** onorando	L,	38,	4
cariss. in Cristo **Padre** e Fratello	L,	41,	2
carissimo ad. **Padre** in Cristo	L,	41,	15

(a) (b) (c)

Fig. 2. One example page (a), the resulting image after cropping and noise removal (b), and after skew correction (c).

tool to recognize and extract ToC from PDF making navigation of book content easier. In order to extract the ToC from printed documents, one solution has been proposed in [6] to support the indexing of documents in a digital library.

In this work we describe a system designed to extract information from one digitized book of concordances. To improve the results achieved by a standard OCR tool we integrate specific methods to extract and recognize specific features of the document (e.g. bold words) and take into account the mutual relationships of context textlines for different entries of concordances.

2 The Proposed Method

To better illustrate the proposed method we show in Fig. 2(a) one example page where we highlight the main components of the structure: the text, indices, and some features of pages, like empty spaces between paragraphs, and noise in the page border. After cleaning the page with suitable pre-processing (Fig. 2(b)) the page is also slightly skewed and is rotated (Fig. 2(c)) to make the subsequent `OCR` process easier.

2.1 Pre-processing

The aim of pre-processing is to improve the quality of document images in order to produce an enhanced page for further analysis [4]. The pre-processing is automatically performed on each page and is composed by four main steps.

Initial Crop. The first naive process is to crop the input image with a specific dimension. The aim is to remove the noise occurring in some page edges and mainly due to the document scanning. Among other items it is important to remove the page number (on the bottom-right of Fig. 2(a)) that could harm the de-skew and segmentation. The final size for each page is 1400×1660. The crop is approximately 200 px wide and 600 px high. The initial image crop is made with a standard projection profile based method. The cropped image is shown in Fig. 2(b).

Noise Reduction. Noise reduction is a filtering transformation, which leads to the removal of most noise. First of all the image is converted in binary [8] and connected components are computed together with their total area. Then, small dotted regions (corresponding to noise) are filtered out.

Skew Correction. Skew correction is a geometrical transformation, which leads to an improvement for the OCR phase. Firstly, the image is projected vertically to get an histogram of the pixels. This process is repeated for various angles. The best angle leads to the maximum sum of the squared differences between consecutive histogram values. Once we obtain this angle, the image is then rotated, as shown in Fig. 2.

Blank Spaces Removal. Blank spaces that may occur between paragraphs are removed to allow the location of a single text block in the page. The result is shown in Fig. 3(a).

2.2 Text Block Finding

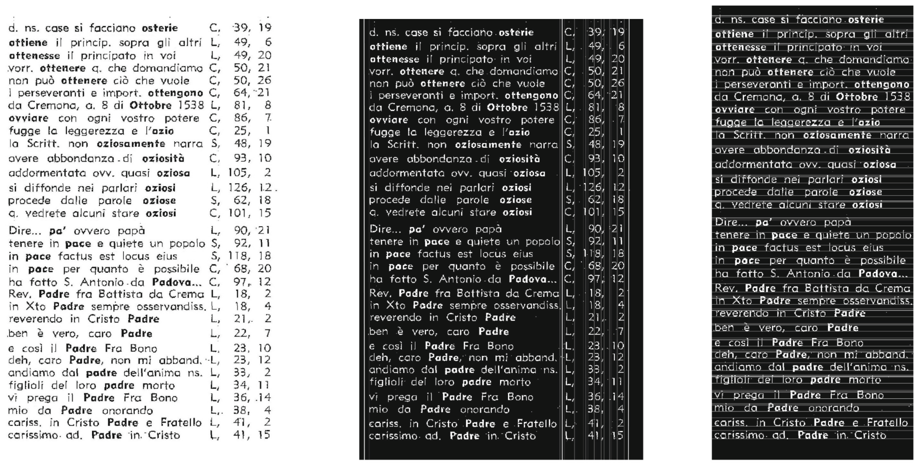
Text blocks are located by using standard techniques that take into account their regular shape. To find a text block in a page, we need the following steps:

1. **Canny edge detector:** that uses a multi-stage algorithm to detect edges in images (the colors are inverted by its execution);
2. **Rank filter:** which is needed to remove the borders. It essentially replaces a pixel value with the median of the ones to its left and right. A border pixel is then easily noticed by their new value;
3. **Removal of possible noise:** finding the contours in the border of the image (contours with a large bounding box) and cleaning everything outside them.

Finally, we look for a crop that is maximizes the number of white pixels inside it and is as small as possible. A standard way to solve this problem is to optimize the *F1 score*, which is the harmonic mean of *Precision* and *Recall* [11].

This is the pipeline used to improve the pages before segmenting the textlines and performing the recognition of text and indexes in each page.

Texline Segmentation. To extract textlines we first convert the image in black and white and its colors are inverted. Then, we compute the mean vector of all rows. For each element of the vector, a comparison between a threshold and



d. ns. case si facciano **osterie**	C,	39,	19
ottiene il princip. sopra gli altri	L,	49,	6
ottenesse il principato in voi	L,	49,	20
vorr. **ottenere** q. che domandiamo	C,	50,	21
non può **ottenere** ciò che vuole	C,	50,	26
i perseveranti e import. **ottengono**	C,	64,	21
da Cremona, a. 8 di **Ottobre** 1538	L,	81,	8
ovviare con ogni vostro potere	C,	86,	7
fugge la leggerezza e l'**ozio**	C,	25,	1
la Scritt. non **oziosamente** narra	S,	48,	19
avere abbondanza di **oziosità**	C,	93,	10
addormentata ovv. quasi **oziosa**	L,	105,	2
si diffonde nei parlari **oziosi**	L,	126,	12
procede dalle parole **oziose**	S,	62,	18
q. vedrete alcuni stare **oziosi**	C,	101,	15
Dire... **pa'** ovvero papà	L,	90,	21
tenere in **pace** e quiete un popolo	S,	92,	11
in **pace** factus est locus eius	S,	118,	18
in **pace** per quanto è possibile	C,	68,	20
ha fatto S. Antonio da **Padova...**	C,	97,	12
Rev. **Padre** fra Battista da Crema	L,	18,	2
in Xto **Padre** sempre osservandiss.	L,	18,	4
reverendo in Cristo **Padre**	L,	21,	2
ben è vero, caro **Padre**	L,	22,	7
e così il **Padre** Fra Bono	L,	23,	10
deh, caro **Padre**, non mi abband.	L,	23,	12
andiamo dal **padre** dell'anima ns.	L,	33,	2
figlioli del loro **padre** morto	L,	34,	11
vi prega il **Padre** Fra Bono	L,	36,	14
mio da **Padre** onorando	L,	38,	4
cariss. in Cristo **Padre** e Fratello	L,	41,	2
carissimo ad. **Padre** in Cristo	L,	41,	15

(a) (b) (c)

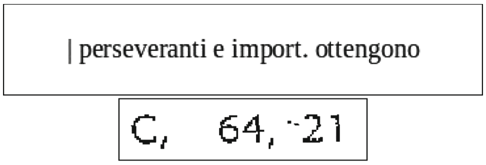
Fig. 3. Left to right, page after removing blank spaces, separation between text and index, identification of texlines.

both the considered item and the next one is made. To perform the textline segmentation we look for the upper and lower border of each text line. We have an *upper* line when the considered item is lower or equal than the threshold and the next one is greater than this threshold; there is a *lower* line when the opposite situation occurs. To illustrate the output of this process, we show the segmentation lines in Fig. 3(c) where *upper* and *lower* lines are drawn in green and white respectively. Even if in this Figure we show the segmentation only for the text part of each page, it is made also for the index part.

2.3 Text Recognition

Text recognition is performed by using `Tesseract` that is an `OCR` engine with support for Unicode and the ability to recognize more than 100 languages out of the box. It is an open source project developed over the past 20+ years at HP and Google and it can be trained to recognize other languages. In this work we mostly used the `Tesseract` version 4.00, except for the index identification where we used the 3.04 version because of implementation reasons. Before using `Tesseract`, we divide the image into two different parts: the text and the index as shown in Fig. 3(b) where the separation is marked with a white line.

In a preliminary implementation of the system we attempted to use the layout analysis algorithms implemented in `Tesseract` to recognize the whole page. Unfortunately, this approach does not provide satisfactory results especially for the index part of the page. To recognize the page content we therefore need to first identify and segment each textline in the main text and then segment each index entry.


| perseveranti e import. ottengono

C, 64, 21

Fig. 4. Example of textline (top) and the corresponding index (bottom).

Main Text. The text extraction is simpler than the extraction of the index. We first compute the distance between closer horizontal *upper* and *lower*, lines. If this distance is above a specific threshold, we hypothesize the presence of a text line. To better identify the text areas we need to enlarge the textline area before recognizing it with the OCR. In particular, for the first and last lines we increase the *upper* and decrease the *lower* positions by 10 pixels, respectively. For the other textlines, these values are modified growing the area half of the difference between the previous horizontal *lower* and the actual horizontal *upper* for the upper side, and half of the difference between the next horizontal *upper* and the actual horizontal *lower* for the bottom side.

To make the `OCR` task easier, we copy the detected text in a new empty image, leaving out some edge pixels for each side. Furthermore, we apply a morphological **dilation** to make the text thicker, using a cross-shaped kernel and apply **Gaussian blur** to smooth the image.

Finally, using `Tesseract`, we convert the image to string. As an additional control, we check if the text length is above a certain threshold because it may happen that some lines are not real text lines, but just noise. A text line sample is shown in Fig. 4 with the respective result extracted by the `Tesseract` OCR engine.

Index. As mentioned in the introduction, in the concordances the index refers to three values, separated by commas. We first have a letter describing the referenced book and three values are possible in this case: [L, S, C] (*Letters*, *Sermons* and *Constitutions*); the page number $\sim [0, 200]$; and the line number $\sim [0, 60]$. The identification method for index lines is analogous to the one used in the text extraction. For better recognition, also in this case the index is copied into a new image, larger than the original one. Since the index structure is regular, we make a vertical segmentation for separating each element of the index: this leads to a better accuracy than performing the `OCR` in the whole line.

For each element, to avoid OCR errors due to the mis-identification of commas as letters or numbers, for each connected component we compute the number of black pixels above half of the index image. If this number is larger than a specific threshold, the component is considered as a letter or number. To improve the recognition, for each element we apply a *Gaussian blur filter*. Since the optimal kernel dimension changes for different index entries, we consider different kernel dimensions in a range between 1 and 19, considering only odd numbers. For each blurred image, we get the `Tesseract` result, taking into account the expected type of output (a letter for the first element and numbers for the following ones). At this point for each blurred image (ten in total) we consider the OCR output: a max-voting process will identify which one has the maximum occurrence that is most likely to be the correct one. There might be some problems:

- the "1s" in the index are not well recognized. We solve this problem computing the distance between vertical *upper* and *lower* of each element: if the distance is below a threshold it will be a 1 and the `Tesseract` process will not be performed. In case we are considering the first element, we change the 1 with an L;
- the distance between related *upper* and *lower* is too large. This means that the textline segmentation failed, not separating the elements in the correct way. In this case wee repeat the same process increasing the segmentation threshold until it reaches a maximum value: in this case we replace the result of the considered element with a blank space;

It may happen, in a few cases, that a text line has no index. In this case we just apply `Tesseract` for the whole line, without separating it in index and text part.

3 Corrections

In this section we will introduce some techniques that have been adopted to correct the OCR output and improve the results.

3.1 Dictionary

This technique is related to word correction regarding the `OCR` performed on the Main Text. To this purpose we consider an Italian dictionary[1] and a list of proper names. In addition, we created a python dictionary featured by bigrams as keys. For each bigram we define four lists that contain:

1. the words in which this bigram appears;
2. the occurrences of each word as detected by the OCR
3. the page referred in the index for the sentence containing the word
4. the line referred in the index for the sentence containing the word.

In order to populate the bigram dictionary we need to clean the `Tesseract` results that present some noise, like random punctuation or special characters. We first replace underscores and apostrophes with blank spaces and remove the other punctuations except for dots and commas, that might indicate an abbreviation. We therefore consider only dots and commas that have a character to their left and a white space to their right, but remove dots and commas that

[1] https://github.com/napolux/paroleitaliane/tree/master/paroleitaliane.

could be an abbreviation, but have to their left a word that is actually in the Italian dictionary.

After recognizing and indexing the whole text we can use the overall information to correct erroneous recognitions. For each phrase, we look only for non-dictionary words that are neither abbreviations or numbers (to avoid to modify correct words, numbers and abbreviations). For each selected word W, we select the words containing some bigrams of W, and compute the *Edit distance* between W and the words. The word with the lowest *Edit distance* is considered as a replacement for W. In case we have more than one word, we consider the one which has the maximum occurrences in the whole document. At this point, we replace the word to be corrected with the found one.

3.2 Punctuation

As already mentioned in the previous section, the OCR introduces some random punctuation while reading the images. What we tried to achieve is to remove this random factor with respect to the grammar rules: considering a punctuation, there will always be a character to its left and a space to its right. However, we tolerate two consecutive types of punctuation if one of them is an abbreviation dot (or an erroneous abbreviation comma when the OCR fails to recognize a dot properly). The random punctuation is then removed replacing it with a blank space. In addition, special characters are removed, except for the asterisks, which are needed for volumes comprehension. In Fig. 5, we can see the results of the two previous corrections. In particular, in blue we highlight the words corrections and in red the punctuation removal.

3.3 Index Correction

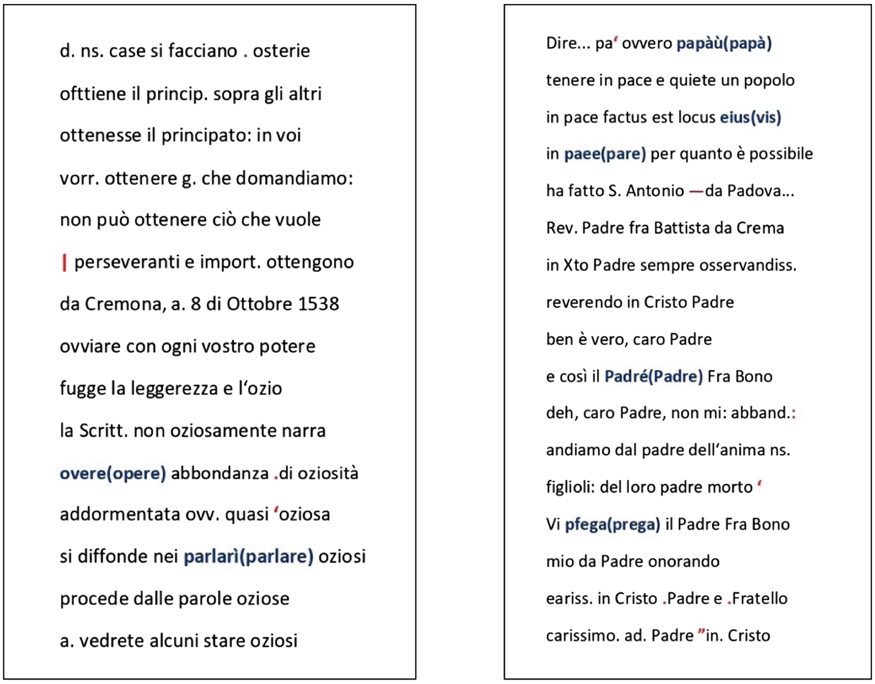
It is clear that for the correction of the index there is little information, if any, that can be provided by dictionaries, since any combination of numbers is in principle possible in an index. On the other hand, in the case of concordances it is not even possible to take into account the consistency of page numbers (that are usually increasing in table of contents).

The unique support to index correction is the main peculiarity of concordance where the context around the indexed word is printed as well. Taking into account this feature it is possible to adjust the page index looking for similar phrases in the rest of the book.

It is easy to notice that often the phrases contain abbreviations and therefore an exact match is not possible. We therefore consider trigram indexing to capture as best as possible similar textlines.

To this purpose, we first compute the trigrams in each textline. We then compute the hash address of the trigram in a space with 2^{16} slots and consider a vector representation of the textline where the vector contains values different from zero in positions corresponding to the trigrams in the textline.

Given this representation, we attempt to correct only the index entry for sentences with a wrong index structure. Let S be one of these sentences, with


d. ns. case si facciano . osterie
ofttiene il princip. sopra gli altri
ottenesse il principato: in voi
vorr. ottenere g. che domandiamo:
non può ottenere ciò che vuole
| perseveranti e import. ottengono
da Cremona, a. 8 di Ottobre 1538
ovviare con ogni vostro potere
fugge la leggerezza e l'ozio
la Scritt. non oziosamente narra
overe(opere) abbondanza .di oziosità
addormentata ovv. quasi 'oziosa
si diffonde nei **parlarì(parlare)** oziosi
procede dalle parole oziose
a. vedrete alcuni stare oziosi

Dire... pa' ovvero **papàù(papà)**
tenere in pace e quiete un popolo
in pace factus est locus **eius(vis)**
in **paee(pare)** per quanto è possibile
ha fatto S. Antonio —da Padova...
Rev. Padre fra Battista da Crema
in Xto Padre sempre osservandiss.
reverendo in Cristo Padre
ben è vero, caro Padre
e così il **Padré(Padre)** Fra Bono
deh, caro Padre, non mi: abband.:
andiamo dal padre dell'anima ns.
figlioli: del loro padre morto '
Vi **pfega(prega)** il Padre Fra Bono
mio da Padre onorando
eariss. in Cristo .Padre e .Fratello
carissimo. ad. Padre "in. Cristo

Fig. 5. Text and punctuation corrections in two pages. In blue words corrections, in red the punctuation removal. (Color figure online)

index I. We first compute the *Jaccard index* between the vectorial representation for S and all those of the other sentences in the document. The *Jaccard index* is used for computing the similarity of sets, that can be expressed using this formula, considering A and B as two hash vectors:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

We take the first 50 sentences (with a syntactically correct index structure) having the highest *Jaccard index*. Afterwards, we compute the *edit distance* between S and the 50 sentences. The sentences S' with the minimum *edit distance* is selected, in case there are more than one sentence we choose the one with the highest *Jaccard index*. Finally, we can fix the wrong index I for sentence S on the basis of the information provided by the index I' for sentence S'. To this purpose we can distinguish three cases where we denote the three parts for index I as $I.L$ (for the letter), $I.1$ (for the first number), and $I.2$ (for the second number):

- If $I.L$ is missing then we copy this value from I' ($I.L = I'.L$);
- If $I.1$ is missing or is out of range then $I.1 = I'.1$;

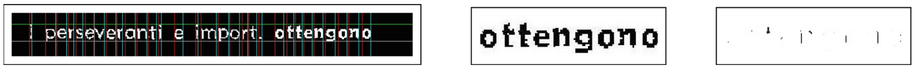


Fig. 6. Word segmentation (left) and identification of bold words: on the center we show one bold word and the results of the erosion on the right.

– If $I.2$ (the line number) is out of range, it is replaced with $I'.2$, but only if $|I.2 - I'.2| > 2$. The latter condition applies to line numbers because similar phrases can have a close line number if some text wrap up.

It is interesting to notice that GPU has been used for index correction because the execution time for the *Jaccard index* computation would have been too long in CPU.

3.4 Identification of Bold Text

For each sentence in the concordances the index word is usually printed in bold. Extracting this feature is important for a good index reconstruction. The idea to find bold words is to first separate each word and then compute a morphological erosion of the word. The erosion is performed using an *elliptical kernel*. Therefore, considering NE as the number of pixels in the not eroded image and E as the number of pixels in the eroded one, the pixel ratio ($PR = \frac{NE}{E}$) will be smaller for bold words because the erosion will leave out some pixels for them with respect to non-bold words that will be master erased.

In Fig. 6 we show the bold word in both not eroded and eroded version. We can notice that in the eroded image of bold words we can still distinguish some black pixels, while for other words the result would have been almost empty. We use three different kernel size for making the algorithm more robust. Eventually, it may happen in very few cases that a phrase contains more than one bold word. In this case, after finding this latter, we check if there are any another pixel ratios, which are close to the one of the bold word. If so, also these words are marked as bold.

Document Transcription. The final step consists in the transcription of all document information to a final *docx* file. This process contains an intermediary step, which produces a *PDF* file as output. This is accomplished in the following way: for each page we transcript the text and indices in an `HTML` file, with a specific structure that divides the page in two main columns, one for the text and the other one for the indices. The index column is also divided in three columns. After merging the *PDF* files, using an online tool, we convert the *PDF* file to a final *docx* one, which can be modified if needed.

4 Experimental Results

In this section we describe the experimental evaluation that is performed on 20 random pages that have been manually annotated. In particular, we analyze the results from two different perspectives:

– recognition of indices and bold words without any correction;
– evaluation of text and punctuation correction.

The results concerning bold words and indices are very good, as shown in Table 1: the algorithms for identifying these two information were already robust and there were no needs for deeper corrections. As described in Sect. 3.3 we implemented one algorithm for indices, but the improvement was very marginal. For each information (bold words, indices, punctuation, text) we define E as the number of errors and N as the total number of that kind of information. The accuracy is then calculated as $Accuracy = 100 - \frac{E}{N} \cdot 100$.

Regarding the text correction we considered 3 different values of a threshold (α) which indicates the maximum allowed *edit distance* for word retrieval. Concerning the punctuation results, the correction leads to an improvement because the algorithm removes most of the random punctuation. Finally, increasing the text threshold, there are more words that can be chosen and also more chance to get an erroneous one. Accordingly, this behavior leads to a lower accuracy for $\alpha = 4$. Using $\alpha = 3$ or 2 instead, we get an improvement of 2%.

Table 1. Accuracy of the proposed method after different corrections.

	Accuracy	
Information	Before	After
Bold	–	98.50%
Indices	–	99.86%
Punctuation	86.06%	89.81%
Text ($\alpha = 2$)	93.44%	95.38%
Text ($\alpha = 3$)	93.44%	95.31%
Text ($\alpha = 4$)	93.44%	93.39%

Table 2. Execution times for different steps.

Phase	Time (s/page)
Pre-processing	11,9
Segmentation	5,85
OCR	154,2
Text correction	2,84
Indices correction	105,2
Bold extraction	6,6

In Table 2 we report the execution times of the various steps. As we can notice the most expensive phases are the OCR (performed by `Tesseract`) and the index corrections. The computer used in these experiments has a CPU Intel i7 7700HQ (Quadcore) and a GPU Nvidia GTX 1050 (4 GB) with 16 GB of RAM and a Samsung SSD 970 EVO Plus.

5 Conclusions

In this paper we presented a system for the recognition and indexing of a concordances book. Despite the good recognition of printed text by means of off-the shelf OCR tools the recognition of structured text containing short sentences with abbreviations and numbers is not perfect. We addressed the problem by considering specific correction techniques like punctuations and bold typefaces. For index correction we considered on the other hand the rendundancy of information of concordances.

For further studies, it would be good to focus also to non-italian words that appear in the document, like Latin ones, that may improve the text accuracy. Another idea is to correct the words depending also on the text context, because we noticed that several `Tesseract` errors concern the distinction between the letters *o* and *a*.

This approach can be extended also to deal with other indexes in books belonging to digital libraries, like table of contents that are the next target of this research.

References

1. Cagni, G.M.: Concordanze degli scritti di S. Antonio M. Zaccaria. Collana spiritualita barnabitica, 4 (1960)
2. Capobianco, S., Marinai, S.: Text line extraction in handwritten historical documents. In: Grana, C., Baraldi, L. (eds.) IRCDL 2017. CCIS, vol. 733, pp. 68–79. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68130-6_6
3. Cesarini, F., Gori, M., Marinai, S., Soda, G.: Structured document segmentation and representation by the modified X-Y tree. In: Fifth International Conference on Document Analysis and Recognition, ICDAR 1999, Bangalore, India, 20–22 September 1999, pp. 563–566 (1999)
4. Gatos, B.G.: Imaging Techniques in document analysis processes. In: Doermann, D., Tombre, K. (eds.) Handbook of Document Image Processing and Recognition, pp. 73–131. Springer, London (2014). https://doi.org/10.1007/978-0-85729-859-1_4
5. Likforman-Sulem, L., Zahour, A., Taconet, B.: Text line segmentation of historical documents: a survey. Int. J. Doc. Anal. Recognit. **9**(2), 123–138 (2007)
6. Mandal, S., Chowdhury, S.P., Das, A.K., Chanda, B.: Automated detection and segmentation of table of contents page from document images. In: 2003 Proceedings of the Seventh International Conference on Document Analysis and Recognition, vol. 1, pp. 398–402 (2003)

7. Marinai, S., Marino, E., Soda, G.: Table of contents recognition for converting PDF documents in e-book formats. In: Proceedings of the 10th ACM Symposium on Document Engineering, DocEng 2010, pp. 73–76. ACM, New York (2010)
8. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979)
9. Read, A.W.: Dictionary, Encyclopaedia Britannica (2016). https://www.britannica.com/topic/dictionary. Accessed 30 Sept 2019
10. Smith, R.: An overview of the Tesseract OCR engine. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, pp. 629–633, September 2007
11. danvk: Finding blocks of text in an image using Python, OpenCV and numpy (2015)
12. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes: Compressing and Indexing Documents and Images, 2nd edn. Morgan Kaufmann Publishers Inc., San Francisco (1999)