Steps in building a transcription technology: deciphering the content of historical romanian documents

Petru Rebeja[1], Eduard Coman[2], Claudiu Marinescu[2] and Dan Cristea[3,1]

[1] University "Alexandru Ioan Cuza", 16 Berthelot St., Iași, Romania
[2] University Transilvania of Brașov, 29 Bd. Eroilor, Brașov, România
[3] Romanian Academy, Institute for Computer Science, 2 Codrescu St., Iași, Romania

Abstract

The paper presents the general lines of a technology aimed to decipher old Romanian documents in the form of uncials and prints, from the Cyrillic into the Latin script. The focus is on two final modules of the pipeline that assembles this technology, one recognizing character shapes and the other sequencing characters which are placed in-between lines by the copyists. The endeavor is motivated by the huge collection of old documents existing in libraries, only very few of them having received the attention of expert linguists.

Keywords

old documents digitization, Cyrillic-Latin transliteration, character identification, character recognition, linearization, deep learning

Introduction

We present in this paper an approach towards building a technology that would help to recover a treasure trove of Romanian culture, still largely unknown to the general public, and also to offer to researchers of the language an intelligent tool for dealing more easily with old documents. The main purpose of this technology is to interpret old Romanian writings written in the Cyrillic script into the Latin script. We believe that our approach, although specifically directed for deciphering Old Romanian documents, could be easily adapted for other languages.

We explain the motivation for this endeavor and the difficulties of the task in this introductory section. In section 2 we present a quick review of the technological line envisioned, then some approaches we consider rather close to ours, and what data we have used to support the training of the modules. Section 3 presents in more details how are characters identified, recognized and linearized and our evaluations referring to the last steps. Some conclusions and discussions are given in the end.

1.1 Motivation

The period of Cyrillic writing in regions of Romania is subject to historical and political circumstances, the use of Cyrillic in writing Romanian coinciding with the use of Slavonic as an official language in the royal courts of Moldavia and Wallachia, but also as a language of orthodox worship in all areas inhabited by Romanians, therefore including also the province of Transylvania. A precise moment when this has happened is controversial [1] [2], but this type of writing was in use at least between the XIIIth century and the half of the XIXth century. The works were mostly church texts, circulating in manuscript form and, only much later, as prints. The lack of rules to impose the grammar and the phonology, as well as the vast territory on which

VIPERC 2023: The 2nd International Conference on Visual Pattern Extraction and Recognition for Cultural Heritage Understanding, September, 25-26, 2023, Zadar, Croatia
petru.rebeja@gmail.com (P. Rebeja); eduard.coman@student.unitbv.ro (E. Coman); claudiu.marinescu@student.unitbv.ro (C. Marinescu); dan.cristea@acadiasi.ro (D. Cristea)

Romanian was used and the documents were produced, with peculiarities of regional pronunciation and foreign influences (mostly Hungarian and Polish) that dictated the spellings, issued a vast diversity of word forms.

Libraries, museums and personal collections include thousands of documents, mostly books, only rather few of them having received the attention of linguists for being transliterated and studied. All this configures a strong need for a technology able to help the interpretation of old Cyrillic Romanian documents, a technology to be put in the hands of scholars (linguists, historians, theologists, geographers), of students, editors and the general public – a category to whom the old writings are now inaccessible.

1.2 The Difficulties of the Enterprise

Since our work is focused on old Romanian texts which are not publicly available, we had to start by putting up a consistent collection of documents, part of them annotated, to form the ground truth and experimenting data. There were no readily-available corpora that we could use. Since our data was bimodal (images of pages and textual transcripts) and it requires fast access and updates, we had to develop a hybrid storage solution to accommodate the entire data set [3].

Annotations on Romanian documents at the level of Cyrillic letters were acquired through a dedicated web application[2], in close cooperation with linguists, whose hints and corrections have been used as design parameters. In some cases, the data has been double checked, but seeing the complexity of the annotation task, we have mainly relied on the high dedication of our colleagues to eliminate the need for multiplying the annotation and performing inter-annotator agreement.

Our collection of scanned pages[3] proved to have varying sources and to be of varying qualities, covering three centuries and a half (from the XVIth to the middle of the XIXth), originating from all Romanian provinces, including pages with defects such as spots, damages or differences of illumination, including palimpsests on which remains of an original writing was still visible, pages scanned at various angles or displaying skews due to thick book spines, including large titles, frontispieces, ornaments and drawn first letters in paragraphs, including marginal writing and interlinear characters, with a wide variation of forms for similar characters and a wide variation of letter sizes, including manuscripts as well as prints, and displaying varying densities of characters per page.

2. A Birds-Eye View of the Technology

2.1 Related Work

We mention only a few systems displaying similarities with our work. The µDoc.tS project [4] [5] is trained to decode Greek, English, German and Finnish. Monk[4] is a tool able to transcribe lines for speeding up the process of indexing a documentation, being used to process, apart from Dutch, also Chinese and Arabic characters. Transkribus[5] is an ABBYY AI-based system offering online and desktop transcribing services for historical documents. It decodes handwriting, even scripta continua, but the documentation does not put in evidence the capacity to recuperate and place in sequence interlinear writing. To the best of the author's knowledge, while some attempts to interpret old Romanian texts are reported [6], they are based on using semi-proprietary technologies. Teiresias [7] is a classification model that associates written archaic Greek symbols to modern Greek letters. Both Teiresias and our system use classification models built from scratch using convolutional layers spliced with max pooling layers, which are then followed by dense layers. However, our model has a smaller number of layers, namely 8 whereas Teiresias uses 18 layers.

[2] as part of the DeLORo project (see Acknowledgments)
[3] kindly offered by the Library of the Romanian Academy
[4] https://www.ai.rug.nl/~lambert/Monk-collections-english.html
[5] https://readcoop.eu/transkribus/?sc=Transkribus

2.2 Methodology

In our vision, the process of interpretation of old Romanian uncials and prints that include interlinear writing is a pipeline having five layers: (i) preprocessing of scanned pages, (ii) character detection, (iii) character labelling, (iv) character linearization and (v) words formation.

Preprocessing means to prepare the image of a page that is received in input by passing it through a sequence of transformations intended to augment the clarity, making the following steps easier, while also squeezing the volume of stored data: binarization, filtering and deskewing. Binarization drastically reduces the storage space of an image, by getting down the memory used to store each pixel from three bytes of memory (colors in the RGB convention), to just one bit (with the meaning of white versus black). Then, to reduce noise and enhance contrast, a Gaussian blur filter is applied, followed by a correction operation that reduces the curvatures of pages. The **character detection** step aims to identify all characters in the image, thus separating them from other types of visual objects. **Character labelling** is a classification process, in which the objects identified in the page as characters receive labels from the set of Latin letters. In the **linearization** step the character boxes identified in the page, those clearly placed in lines of text as well as those written in-between lines, are linearly ordered, according to the positions of their bounding boxes in the original image. Then, a string of Latin letters is formed by picking their transcribed labels. Finally, in the obtained string **word boundaries** are placed, by exploiting either linear distances between the original boxes (where they are) or linguistic criteria (occurrence in a dictionary) or both.

2.3 Collecting Data in Support of our Approach

The dataset we used for this work make up a corpus of 52 Cyrillic Romanian historical texts, totaling 16,864 images of original pages (both in print and uncial form) over a span of three and a half centuries. This collection, called the Romanian Old Cyrillic Corpus (ROCC) [3], has been organized into a database, in which the documents also contain metadata and annotations. The metadata includes information such as: title, author, genre, place of publication, year, and quality of the original and scanned copy. The inclusion of metadata significantly increased the value of the dataset by providing contextual information that, at later steps, could help in the interpretation and deciphering of the documents. Having access to actual pages from old documents (see Figure 1 for some examples), as opposed to simulated data or modern transcriptions, allows us to address real-world challenges such as variations in script styles, text degradation over time, and scan quality effects. It provides an opportunity to design solutions that are robust and versatile, able to cope with a wide range of conditions.

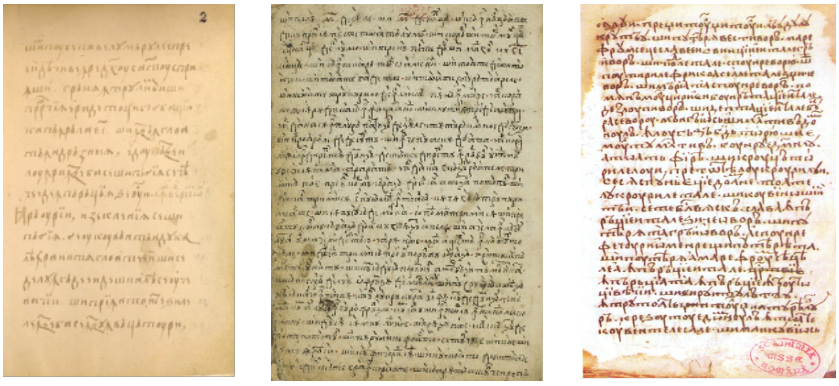


Figure 1: Examples of pages from the ROCC dataset

ROCC also includes annotations operated manually by members of the DeLORo project. At the moment of this article, the statistics displayed by the online annotation frontend shows 191,622 characters, 7182 rows of text, 5810 modifiers, 1972 letters outside rows and 186 titles.

3. The Main Steps of the Technology

3.1 Binarization

Binarization is the preprocessing technique that converts a color image into a black and white one, effectively removing noise and other extraneous details and facilitating further processing by machine learning models (see Figure 2). This is done in two steps: filtering followed by the effective binarization. In the filtering phase the value of a central pixel is recomputed to take in consideration the neighboring ones, while in the adaptive binarization step a context-dependent threshold value is fixed and all pixels below this threshold are turned black (representing the text) and all pixels above the threshold – white (representing the background).

The adaptive type of binarization we have used accounts for local illumination conditions in different regions of the page, thus helping the model to focus more easily on text characters and to recognize them.

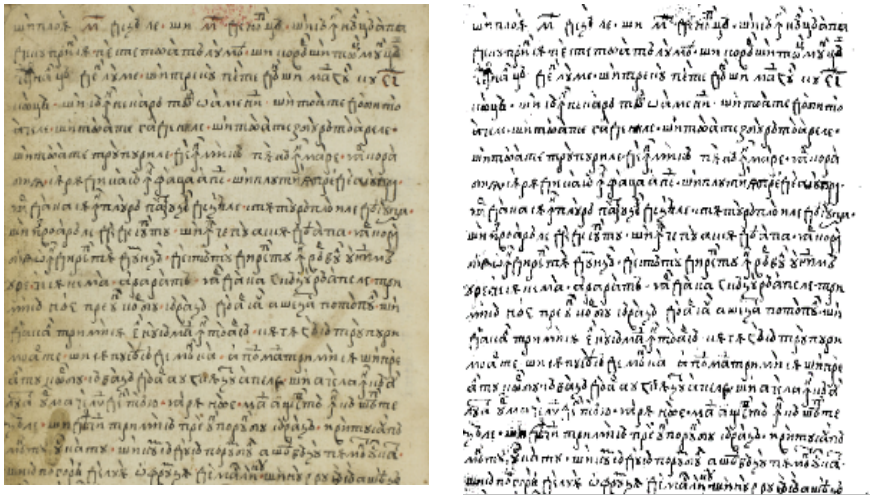


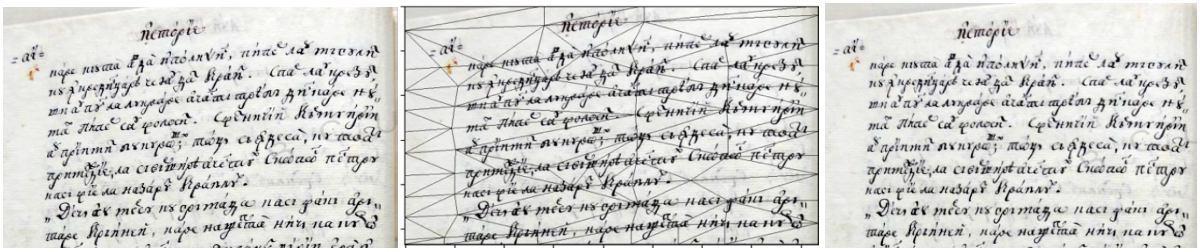
Figure 2: Example of applying binarization to a page: original (left) and the result (right)

3.2 Deskewing Page Curvatures

In the field of digital transformation, a recurring challenge that arises when trying to digitize physical documents, especially books that have a thick book spine, is the appearance of distortions introduced during the scanning process. In most cases, these distortions are manifested as curved lines of text, actually lines deviating from the parallelism they should normally have with the top and bottom edges of the pages. The motivation for removing page curvatures comes from the goal we have, which is to obtain sequences of deciphered letters that could be read linearly. Since the linearization process should finally follow the lines of text, one important step is the detection of the text rows. These are horizontal geometrical lines approximating the maximum density of pixels. It follows that it is stringent that the lines of text be straight and horizontal.

The deskewing process has been thoroughly described in [8] and consists of two major steps: detecting the curved lines of text in the input image and rectifying the image to make these lines

straight and horizontal[6]. For detecting lines, on the inverted image, a Gaussian 2D low-pass filtering is first applied, with a kernel much more aggressive on the horizontal direction, to exploit the a-priori expectation that text lines are oriented more or less horizontally; then the line blobs are detected by using a ridge following strategy to obtain a set of connected small segments, which approximately indicate the middle of the text lines. In the image rectification phase, the image is warped in such a way as to transform the detected connected segments into horizontal lines, by aligning horizontally the segments that approximate the highest density of pixels in lines, with the regions of pixels around them being deformed accordingly. For the actual image dewarping, a mesh decomposition with spline transformation has been applied. An illustrative example of a fragment of a page, before, in the middle of the process and after deskewing is shown in Figure 3.



(a) (b) (c)

Figure 3: The deskewing process: a fragment of a page on the initial scan (a); the mesh decomposition (b); after deskewing (c) – from [8]

3.3 Detecting Characters in Pages

In the DeLORo project [3], more types of graphical objects have been manually annotated on the original collection of scanned pages and included in ROCC: lines of text, letters (characters), drawn initial letters, titles, frontispieces, modifiers, marginal letters, interlinear letters, reference marks, accolades, ornaments, etc.). Out of all these categories of graphical objects, in this paper we are concentrated solely on characters, being them in lines or in-between lines. The process goes through two steps: letter detection (also called identification) and labelling (also called recognition).

In the letter identification phase, each letter is delimited by a rectangular box. This is defined by the horizontal and vertical coordinates of the upper-left corner, a width and a height. This way, each letter is isolated and prepared for the next step which involves classification.

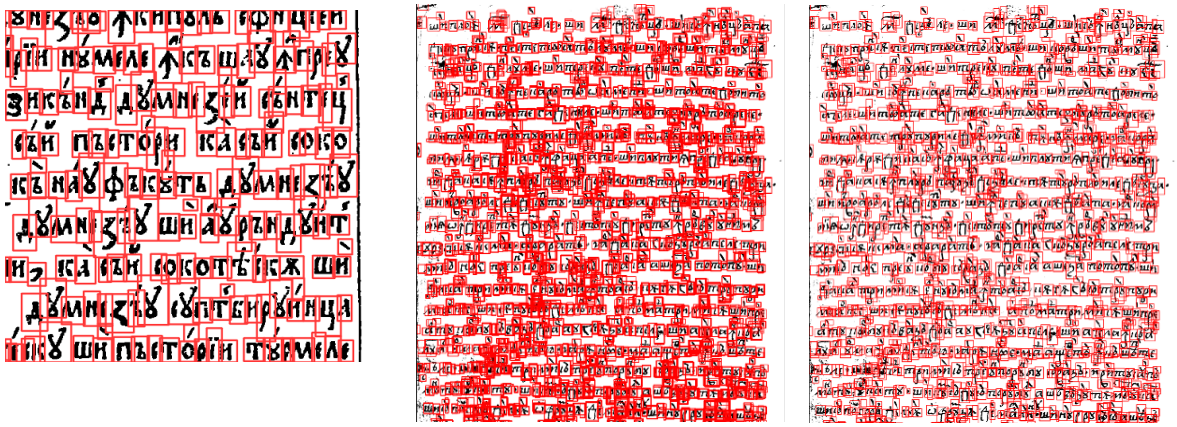
In order to achieve character detection, we opted to segment the page image into numerous windows that, on one side, overlap and, on the other, span the entire size of the image. In [8] the character detection performance of a YOLOv5 model [9] trained on whole pages was compared with one trained on squared windows cut down from pages. The windows segmentation approach showed a significant improvement, with the average mean precision (mAP) at an intersection-over-union (IoU) threshold of 0.5 increasing from 0.82 (for full images) to 0.99 (for segmented images).

The window segmentation methodology, proposed in [10], divides images into windows of size 500 by 500 pixels. The training set consisted of 1,000 windows containing 140,324 annotated character boxes, while the validation set included 500 windows, with 48,359 annotated character boxes. By making the windows to slightly overlap we capture information (characters) that might otherwise be split or clipped at the windows' edges, ensuring thus that no characters are lost during this process.

Each of the windows produced by the segmentation process is then processed independently for character detection. Once recognized in the windows, the characters detected are recomposed to form an integral page, preserving their spatial positions in the original image. To eliminate

[6] https://github.com/nikcleju/RowRectification

fragmented characters and remove duplicates, the detected boxes are then passed through a purging algorithm, which eliminates duplicates and maintains the most reliable detections (best candidates) for each character (see Figure 4a).



(a) (b) (c)

Figure 4: Characters isolated in boxes: on a fragment of a page (a); on a whole page, before running the pruning and consolidation processes (b); and after (c)

Duplicate detections in separate windows are eliminated by comparing the sizes of intersecting bounding boxes and their confidence scores, as assigned by the YOLOv5 model to each detected character. This confidence score reflects the model's certainty in detection, with higher scores indicating greater confidence. Then, after going through more heuristics to decide the best candidates for retaining, while pruning the rest, our experiments highlighted the following behavior: when two detections totally intersect (one bounding box is included into another) or their intersection is partial but significant (the IoU metric is above a threshold), they are considered to be duplicates and the one displaying the highest product between the area and the confidence score is retained; when the IoU is under the threshold, we implemented a consolidation process, which ensures that the intersecting characters are recognized as separate entities, by considering factors such as character detection model confidence, bounding box sizes, and spatial relationships between detected boxes; see Figure 4 (b) and (c).

The process of recombining the segmented image and reconciling the results from overlapping windows, although complicated, is vital for accounting the success of the next steps.

3.4 Classifying Characters

The following step focuses on the transliteration of identified Cyrillic characters into their Latin equivalents. Character classification consists of assigning a label to each box. The label is represented by a Latin letter or a number, if the object is clearly recognizable. An additional dimension of complexity is linked to the fact that some characters can be drawn in more than one way (see Figure 5 for some examples), depending on the region of the author and the historical period. The many-to-one choices are solved by assigning one Latin label to more Cyrillic character shapes. Inversely, the one-to-many choices are usually nuanced by the context. In the situation when the character is not recognized or there is uncertainty about its identity, the character will be labelled with the *$* symbol. The approach presented in the present work involved implementing several deep learning models designed to handle these complexities efficiently, as well as adjusting a pre-trained model to suit our case. In our experiments, we tested several model architectures and data pre-processing techniques, finally opting for a model that demonstrated the best results.

Our first implementation is a variation of a Convolutional Neural Network (CNN), a class of deep learning models particularly effective in image recognition tasks, consisting of multiple

layers of convolutional filters that learn hierarchical representations of input data and extract meaningful features from images, capturing spatial patterns at different scales.

Our data set comprised approximately 28,000 images, each representing a character written by hand (in uncial writing) or a printed one. The output should decide one (or more) out of 24 distinct character classes, the training set containing approximately an equal number of instances, thus ensuring that the model does not favor any particular decisions. The images in the dataset, collected manually in DeLORo, are split as such: 40% training set, 40% validation set and 20% test set. Moreover, each character box is normalized to a square of 28x28 pixels.

(a) (b)

Figure 5: Letters having different forms: letter *i* (a) and letter *ă* (b)

Inspired by LeNet-5 [11] [12], our first architecture incorporates convolutional and pooling layers to capture relevant spatial features from the input image. We use two convolutional layers with filter sizes of 3x3 and 2x2 and a 2x2 max-pooling layer after each convolutional layer.

Another direction to improve the results has been to implement a range of strategies (data augmentation, early stopping, hyperparameter tuning), which not only optimized the learning process and increased the model's ability to generalize, but also effectively prevented overfitting. The implementation of these changes marked a key achievement: breaking the 0.95 threshold for model precision for the first time.

As said already, the character identification phase places bounding boxes around characters and this process uses a binarized image, by applying adaptive thresholding, which exploits the variations in quality and illumination in different regions of the page. Once the characters have been identified on the whole page, to effectively classify the shapes inside boxes, we could come back to the original image and apply binarization at this local level. The experiments to gain more precision went on in two directions: choosing different types of binarization methods and applying variations in hyperparameters, such as the batch size, the number of epochs, and the learning rates. Table 1 shows the results for four types of binarization methods. As can be seen, the best results (0.9565) are obtained with a grayscale binarization.

Table 1: CNN experiments (precision)

	Grayscale	Fixed threshold	Otsu's method	Adaptive threshold
Different filter sizes	0,9555	0.9171	0.954	0.9491
Add conv. layer 128 filters 3x3 kernels	0.9461	0.9096	0.9368	0.9385
Initial architecture	**0.9565**	0.9279	0.943	0.9526

Then, using the best model, we have considered a different division for the three components of the dataset (80/10/10 instead of 40/40/20), which yielded a slight improvement over the 96% threshold for precision: 0.9645.

To further raise the quality of the classifier, we also explored an alternative approach, by fine-tuning a pre-trained model. Fine-tuning means taking a pre-existing model, trained on a large data set, often for a different task, and adapting it to our specific problem. This approach uses the knowledge and feature extraction capabilities already learned by the pretrained model, which can lead to improved performance and reduced training time.

ResNet, short for Residual Network, is a deep convolutional neural network architecture that has gained significant popularity in computer vision tasks. It was introduced to address the vanishing gradient problem encountered when training very deep networks [13]. ResNet introduces residual connections, or skip connections that allow the network to learn residual functions, i.e., the difference between the input and the output of a layer. This helps to address the problem of degradation caused by increasing network depth.

To tune ResNet for our character classification task, we used the ResNet50 variant[7]. The model was pre-trained on the ImageNet dataset [14], which contains a large number of images from a wide range of classes. In our implementation, we replaced the final layers of the model to suit our classification problem. We added a flatten layer, a dense layer with ReLU activation function and a final dense layer with softmax activation for the output, while the pooling layer helps to summarize the most essential features learned from the previous layers. We compiled the model with the 'categorical cross-entropy' loss function and the Adam optimizer, which has proven effective for many deep learning tasks [15]. By using the ResNet50 variant, initially trained on the ImageNet dataset, we took advantage of a wealth of diverse visual knowledge that proved beneficial in improving the overall feature extraction capabilities in our specific task.

The fine-tuned ResNet50 model not only that matched well on our classification problem, but also significantly outperformed other architectures we have used during the experiments. With ResNet50 the classification accuracy performance exceeds the 0.97 threshold (0.9714) for the first time, highlighting the substantial improvement in the model's ability to correctly identify and transliterate a wide range of Cyrillic characters.

Finally, in a last experiment we adopted the VGG network (short for Visual Geometry Group) – a convolutional neural network architecture well known for its efficiency in computer vision tasks. Like ResNet, VGG was designed to combat the vanishing gradient problem [16]. It includes two convolutional layers with increasing filter sizes (32 and 64), a choice which allows the model to progressively capture more and more complex traits and patterns present in the characters. The VGG architecture is notable for using small and dense convolutional filters, especially 3x3 filters, which allow the network to learn complex patterns with a small number of parameters. In addition, VGG uses pooling to reduce the spatial dimension of the input volume while increasing the depth, helping to extract meaningful features.

Figure 6: A fragment of a decoded page

For our problem, we chose to use the VGG16 model, a variant of the 16-layer VGG architecture, due to its efficiency in image recognition tasks. The model was pre-trained on the same ImageNet dataset used by us to pre-train the ResNet50 model. To adapt VGG16 to our character classification task, we modified the upper layers of the model to suit our needs. Following the convolutional layers of the base model, we added again a flatten layer, which transforms the input tensor into a one-dimensional vector, thus allowing connections with the dense layer placed in its tail: a dense layer with 256 nodes. A ReLU activation function was also added to introduce non-linearity, helping to learn complex patterns [17]. Finally, we added a final dense layer with the number of nodes equal to the number of classes and a softmax activation to generate classification probabilities. We compiled the model with the same categorical cross-entropy loss function and

7 https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758 (accessed: 17 April 2023)

the Adam optimizer. The VGG16 model performed impressively in character recognition, achieving an accuracy of 0.9722, thus surpassing for the second time the threshold of 0.97 and proving once again the high potential of pre-trained models. Figure 6 shows a fragment of a decoded page in which the optimum architecture/parameters have been used. The Cyrillic character boxes are marked with letters in the Latin alphabet.

Up to this point in our research, as seen, the best results obtained for the precision of character recognition shows an error rate of less than 3%, which means that one character out of 33 could still be erroneous. If this would be the only source of errors, considering an average length of a word of 5 characters, this result means that one word out of 6 could possibly be misspelt.

In the following, starting from the results obtained thus far, we foresee two ways to continue to raise the quality of the character recognition task: by improving the CNN model (making modification to the architecture and playing with hyperparameters) and by adopting a post-processing phase that would use geometric and lexical information to form words and correct them. About this type of processing we will talk in the next subsection.

3.5 Putting Characters in Order

In old documents, especially in manuscripts as those with uncial writing, characters are often placed interlineally. This reflects the copyists' acts of successive corrections applied to the original hand written document. A linearization algorithm was proposed that arranges the boxes in a left-to-right sequence based on their position with respect to the rows of text and the sequence of rows within the page. Any box that is positioned between rows (an interlined box) should be placed in one of the adjacent rows. The evident assignment is for characters placed above the first row and under the last one.

The idea of this processing phase is to assemble sequences of letters from the geometric arrangements of characters, once identified and recognized in the page. Most of the time, boxes of characters have an evident location inside rows, but, when placed interlineally, the technology should correctly find where to insert them in rows.

First, the position of text rows must be determined on a vertical axis of the image. As in the previous phases, the binarized image helps a lot to distinguish between text and background. Then, like in the deskewing operation, a Gaussian filter with a horizontal elliptical mask is applied to smooth the image of the text, and the horizontally projected histogram of the pixels is calculated on a vertical up-down axis, this giving a graphical representation of the density of pixels in the image. Finally, the peaks in the histogram (local maxima) are considered to indicate the positions of the rows. Figure 7 shows different phases of this process.

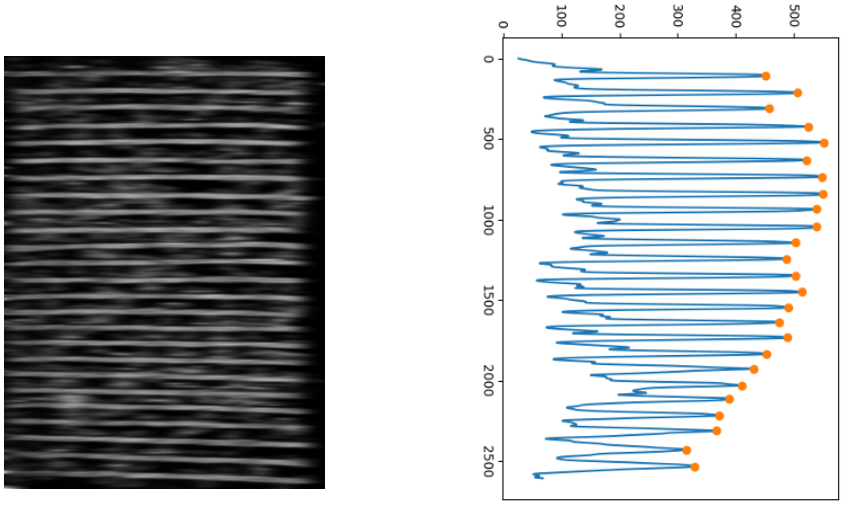


Figure 7: Detecting the position of lines in a page. (a) - the page is inverted and a horizontal Gaussian filter is applied; (b) - the vertical positions of rows are considered to be in the local maxima of a horizontally projected histogram

Character sequencing in a page depends on three decisions: (1) the sequencing of the rows on the vertical axis of the page; (2) the row each character's box should be assigned; (3) the relative positions of character boxes in a row.

The answer to question (1) is simple: as given by the vertical coordinates of the histogram maxima.

After trying more solutions, the answer to (2) is, for the time being, implemented as a simple heuristic in which more space is left above the rows then under them (in a ratio 0.65/0.35, which means that 0.65 of the distance between two rows is allocated for boxes whose vertical coordinates of their geometric centers are above the vertical coordinate of a line, and 0.35 – for boxes placed under – see Figure 8). We are aware that this solution has its drawbacks. A more advanced solution, to be tried in the future, would be to assign boxes to rows based on the centers of gravity of the pixels inside boxes and not to their geometric centers.

To answer (3), the simplest approach would be to sort boxes by the horizontal coordinates of their geometrical centers. In reality, a simple geometric sequencing could be misleading in many cases of interlineally placed characters. For this reason, to which can be added complications triggered by imperfections in the pipeline described so far (characters not identified or identified with a low confidence, false characters induced by spots or trails of large characters, which have misled the segmenter), it became evident that the final decision on the output should be let on the shoulders of a module applying lexical criteria. Indeed, the final output of our technology cannot be a sequence of characters, but a sequence of words. The same type of linguistic knowledge that we invoke here acts in the mind of a human expert when deciphering an old text. It is common knowledge that knowing the language helps enormously in deciphering a text. Within the same paradigm should be treated issues introduced by missing or false white spaces, abbreviations, obsolete word forms or totally unknown words.

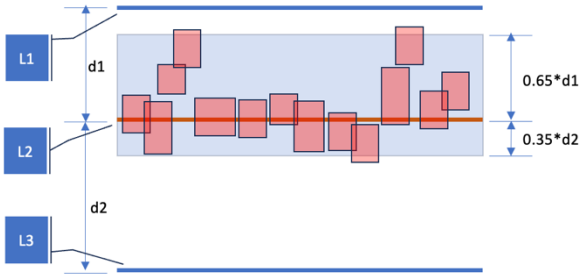


Figure 8: The heuristics for attachment of character boxes to lines

Till now, we have only tested the baseline heuristic of ordering characters in lines based on the horizontal coordinates of boxes' centers. To eliminate the errors caused by the previous steps (character segmentation and labelling) the evaluation should use ground truth in which these previous steps are not present. The lack of this kind of manually annotated data obliged us to execute tests on only a few pages, segmented and labelled ad-hoc, for which we also had expert Latin-script transcriptions. After withdrawing white spaces from the transcription (for evident reasons, as spaces are not noticed by the character segmenter), the evaluation computed the BLEU score [18] – adapted to count characters instead of tokens. We report here the best value obtained: 0.933999.

4. Conclusions and Further Steps

We have discussed in this paper the main solutions that compose the technology of automatically transcribing old Cyrillic Romanian documents, handwritten in uncials or printed, in the Latin script. Part of the solutions proposed are tested, others remain to be implemented. Overall, the technology composes a pipeline of interchangeable modules that will receive scanned pages of old Cyrillic Romanian documents and will output sequences of words drafted in the Latin

script. The vision is that all modules be replaceable, allowing for improvements, either induced by changes in their internal structure or by the addition of more training data.

However, important issues yet remain to be treated. We enumerate here only some:

- some Cyrillic glyphs have more Latin interpretations; therefore, their transliteration should be linked to the context of occurrence. The issue (known as interpretative transcription) has been theoretically treated [6], but a reliable technology that implements it is yet to come;
- we still lack a global evaluation of the pipeline; this could be obtained by deciphering pages for which an expert transcription already exists, as are the critical editions. As suggested in the previous section, the BLUE score could be used for that, comparing the output of the pipeline against the transcribed texts;
- we also foresee the possibility that the pipeline will need to process documents written in the *transition alphabet*, used in the middle of the XIX century on the Romanian territory, which is a mixture of both Latin and Cyrillic characters. There are studies showing how to discriminate between different scripts used in a document, e. g. [19]. In a future development we will certainly have to adopt such techniques capable of classifying individual characters regardless of whether they are Latin or Cyrillic. Having developed such a model will allow us to make the processing pipeline generic, thus, at the expense of more processing cost, to abstract the language it is able to decode.

A number of applications could be issued from the technology described here. Here are some: (a) a tool capable to search the scanned pages of an old document by answering to a keyword inputted by the user, without decoding the whole document; this would involve a back transliteration from Latin Romanian (of the input keyword) into Cyrillic Romanian and an ad-hoc generation of character shapes that copy the writing style or the print of the document; then, a number of forms thus produced would be compared against the document in search for visual matches; (b) digitization of old documents could be paired with their transcription and the automatic recuperation of the word forms contained; this, in time, would help to inventory the old language diachronically (on periods of 50 years, for example) and synchronically (on regions and provinces of the actual Romanian territory).

Acknowledgements

Part of this work has been done as part of the project DeLORo – Deep Learning for Old Romanian, PN-III-P2-2.1-PED-2019-3952, no. 400PED, 2020-2022, official pages at http://deloro.iit.academiaromana-is.ro/. We thank the members in the project from the Institute of Computer Science and the Institute of Philology in the Iași branch of the Romanian Academy and from the Faculty of Mathematics and Computer Science of the University of Bucharest.

We thank the Library of the Romanian Academy in Bucharest for offering scans of old books and metadata.

We thank assoc. prof. dr. Roxana Vieru and her students from the Master in Paleolinguistics, Faculty of Letters, “Alexandru Ioan Cuza” University of Iași, who contributed with hundreds of hours of manual annotation to acquire data used to train the technology.

References

[1] P.P. Panaitescu (1965). Începuturile și biruința scrisului în limba română, București, Editura Academiei R.P.R.

[2] I. Gheție (coord.) (1997). Istoria limbii române literare. Epoca veche. (1532-1780). București, Editura Academiei Române, 1997, 496 p.

[3] D. Cristea, C. Pădurariu, P. Rebeja, A. Scutelnicu, and M. Onofrei (2021). Data Structure and acquisition in DeLoRo - A Technology for Deciphering Old-Cyrillic-Romanian documents. In: Proceedings of the 16th International Conference "Linguistic Resources and Tools for Natural Language Processing", online, 13-14 December, “Alexandru Ioan Cuza” of Iași Publishing House, ISSN 1843-911X, pp. 59-74.

[4] I. Pratikakis (2021). Accessing Greek Historical Handwritten Documents Using the μDoc.tS Platform (DeLORo -> DeLOGr). In Petru Rebeja, Mihaela Onofrei, Dan Cristea and Dan Tufiș (eds.) Proceedings of the 16th International Conference "Linguistic Resources and Tools for Natural Language Processing", online, 13-14 December, "Alexandru Ioan Cuza" of Iași Publishing House, ISSN 1843-911X, pp 3.
[5] L. Tsochatzidis, S. Symeonidis, A. Papazoglou and I. Pratikakis (2021). HTR for Greek Historical Handwritten Documents. Journal of Imaging 7(12):260, 2021. https://doi.org/10.3390/jimaging7120260.
[6] A. Colesnicov, E. Boian, C. Ciubotaru, S. Cojocaru, L. Malahov, L. (2014). Digitizarea, recunoașterea și conservarea patrimoniului cultural-istoric. In Akademos, 32: 1, 61-68.
[7] Berlino, A., Caroprese, L., Mirabelli, G., & Zumpano, E. (2020). Teiresias: a Tool for Automatic Greek Handwriting Translation. In A. Amelio, G. Borgefors, & A. Hast, Proceedings of 2nd International Workshop on Visual Pattern Extraction and Recognition for Cultural Heritage Understanding co-located with 16th Italian Research Conference on Digital Libraries {(IRCDL} 2020), Bari, Italy, January 29, 2020 (pp. 34–45).
[8] D. Cristea, N. Cleju, P. Rebeja, G. Haja, E. Coman, A. Vasilescu, C. Marinescu, and A. Dascălu. (2023 – in press). Bringing the Old Writings Closer to Us: Deep Learning and Symbolic Methods in Deciphering Old Cyrillic Romanian Documents, in Memoirs of the Scientific Sections of the Romanian Academy.
[9] J. Redmon, S.K. Divvala, R.B. Girshick, and A. Farhadi (2015). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788.
[10] D. Cristea, P. Rebeja, and C.C. Pădurariu (2022). Applying YOLOv5 Learning in Detecting Old Cyrillic-Romanian Characters. In: Proceedings of the 17th International Conference "Linguistic Resources and Tools for Natural Language Processing ", Chișinău and online, 10-12 Nov. 2022, ISSN 1843-911X, pp. 115-122.
[11] Y. LeCun, L. Bottou, Y. Bengio, Y. and P. Haffner (1998). Gradient-Based Learning Applied to Document Recognition, Proc. IEEE 1998, 86, 2278–2324.
[12] Y. Bengio and Y. LeCun (2015). 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.
[13] K. He, X. Zhang, S. Ren and J. Sun. (2016). Deep Residual Learning for Image Recognition, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 770-778, doi: 10.1109/CVPR.2016.90.
[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei (2015). ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision, 115, 211-252. https://doi.org/10.1007/s11263-015-0816-y.
[15] D.P. Kingma and J. Ba (2014). Adam: A method for stochastic optimization, ICLR (poster).
[16] K. Simonyan and A. Zisserman (2015). Very deep convolutional networks for large-scale image recognition, in Yoshua Bengio and Yann LeCun (eds.) Proceedings of the International Conference on Learning Representations, in 3rd International Conference on Learning Representations, San Diego, CA, May 7-9.
[17] V. Nair and G.E. Hinton (2010). Rectified linear units improve restricted Boltzmann machines, in Proceedings of the 27th International Conference on Machine Learning, Haifa, 21 June, pp 807-814.
[18] K. Papineni, S. Roukos, T. Ward and W.-J.. Zhu (2002). BLEU: a Method for Automatic Evaluation of Machine Translation, in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July, pp. 311-318.
[19] Brodić, D., Amelio, A., & Milivojević, Z. N. (2015). Characterization and Distinction Between Closely Related South Slavic Languages on the Example of Serbian and Croatian. In. Computer Analysis of Images and Patterns: 16th International Conference, CAIP 2015, Valletta, Malta, September 2-4, 2015 Proceedings, Part I 16 (pp. 654–666).