Towards a Bulgarian Historical Newspaper Corpus — Construction of a Reading Order over the Text in Searchable PDFs

Nikolay Paev[1], **Stefan Marinov**[1], **Ivan Kratchanov**[2], **Petya Osenova**[1], **Kiril Simov**[1]
[1] Artificial Intelligence and Language Technology
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Bulgaria,
{nikolay.paev, stefan.marinov}@iict.bas.bg, {petya, kivs}@bultreebank.org

[2] National Library "Ivan Vazov" - Plovdiv,
Bulgaria,
ivankra@gmail.com

Abstract

The first task in the process of preparing historical newspaper corpora is to determine the reading order of texts. These texts are often extracted from searchable PDFs produced by some OCR software. In this paper we present an algorithm for generating the original reading order of the text blocks selected from the corresponding PDF. We also performed a tuning of the algorithm parameters. The optimization provides an improvement of 10 %.

Keywords: Bulgarian Historical Periodicals, OCR Models for Bulgarian, Large Language Models for Combining Segments of Bulgarian Texts

1. Introduction

Old periodicals are a source of information to support research in many areas of social sciences and humanities (SS&H) —- ranging from linguistics and literature to history, ethnography, journalistic studies, etc. The research in SS&H is in the stage of its active digitization period, in which physical artifacts in archaeology, libraries, archives, museums, art galleries, and others are converted into digital objects via audio and video recording, scanning and OCR processing, 3D scanning and modeling. In parallel to digitization of the artifacts, methods for linguistic processing have also been developed. These methods provide new perspectives on the existing research in SS&H, such as natural language processing services, AI technology, etc. In this paper, we present the first steps in the digitization of old Bulgarian newspapers in the period from the reinstatement of Bulgarian state (1878) to the middle of twentieth century.

The first step refers to detecting the text in the document with an OCR system. During the OCR-ing the important task is to ensure a very good quality with respect to the period of publishing, and to the typography of the newspaper edition — the fonts, selected font sizes, line lengths, line spacing, letter spacing, and formatting of the page. All these factors appear to be very important since the included period is one of great dynamics in the area of publishing, technology, and typographic style in Bulgaria. Also, the standards of spelling have not yet been well-established. From 1870 to 1945, eight spelling forms were proposed and applied in various ways – either officially, or non-officially. These are:

- the Marin Drinov's spelling
- the spelling project by the philological committee from 1893
- the spelling project from 1895
- the Ivan Vazov's spelling system from 1899
- the Drinov-Ivanchevski spelling system from 1899
- the Omarchevski spelling system (1921 - 1923)
- the spelling project of the Historic-philological branch of the Bulgarian Academy of Sciences (1899)
- the Tsankov's spelling system (1923)
- the contemporary system (1945)

The most prevalent spelling system until 1945 (with small interruptions) remains the Drinov-Ivanchevski one from 1899, which to great extent adheres to the traditional/etymological spelling. The linguistic phenomena that usually show heterogeneous features across various spelling systems are as follows:

- *phonetic*: changing of *ya* to *e*; order of the assembly of a *shwa* and *r* or *l*; reduction of the open vowels to non-open ones; the usage of the small *er* at the end of the nouns.

Proceedings of the First Workshop on Creating Interoperable Corpora of Historical Newspapers (PressMint 2026), pages 56–64

- *grammatical*: long and short forms of the definite article in masculine singular nominals; deverbal nouns ending in *-ne* and *-nie*; vocative; pronouns.
- *other*: small and capital letters; spelling of foreign words; spellings with a dash; punctuation.

After 1945 Bulgarian lost some letters, among which the yat, the ers in the end of the words, the nasal 'big nosovka'. Also, full and short articles started to be used in masculine to mark the subject and oblique argument positions, respectively.

To handle these problems, we retrained the OCR model for the peculiarities of each newspaper. The OCR provides the recognized text in a formatting that resembles the formatting of the physical appearance of the scanned documents. In the case of newspapers, the formatting includes various types of information —- titles, date, publisher information, columns of text (containing articles, advertisements, notes), images, page numbers, etc. The OCR-ed result represents this information in the form of separate text blocks with coordinates related to the respective sheet of paper.

After the formal division, our next task is to interpret the content structurally - detecting the page layout, the titles, the paragraphs, the articles, the advertisements, and others. This appeared to be a very non-trivial task. For example, different editions of periodicals have almost arbitrary page layouts, which change on each page. The publishers tried to accommodate as much text from different articles as they could on the first page and also to fill all page spaces with some information. They also employed various layouts which changed the reading order of articles in many ways. All these issues led us to create complex algorithms to extract and then order the text in the complete articles.

The structure of the paper is as follows: the next section focuses on the related work on newspaper article extraction. Section 3 describes the preparation of the dataset with searchable PDFs. Section 4 introduces the processing of the dataset in order to define a reading order over blocks of text. Section 5 presents the evaluation of how well the reading orders are generated. Section 6 outlines the manual post-editing framework. Section 7 concludes the paper.

2. Related Work

Here we present a non-exhaustive list of related works that show good practices for other languages.

In Bourne (2025) Pixtral 12B (OCR using an image-to-text model) is applied to the Nineteenth Century Serials Edition (NCSE) archive, containing about 84,000 pages from six British periodicals. The process includes: automatic detection and post-processing of the layout (bounding boxes) and subsequent text processing. We deal with the same process, using OCR trained/prepared for Bulgarian and for the specific fonts of the printed publications. However, in this paper, we pay more attention to the post-processing of the text, as it is inevitable, but still directly dependent on the quality of the OCR.

Ding and Huang (2014) examine the digitization of historical newspapers in China, focusing on the production of PDF files and the practices used in projects such as the DaChengLaoJiu database, the digitization of Dazhong Daily, and solutions by Beijing companies. They extract text from PDFs and then try different OCR approaches. Not surprisingly, the same difficulties with the extracted text have been encountered as ours, due to the complexity of the OCR processing. They recognize the need for human verification and editing. We also became aware of the necessity for human intervention. Thus, we propose an approach to reduce the amount of human labor required and try to optimize the editing process.

Schultze et al. (2025) presents the Chronicling Germany Dataset – the largest currently annotated corpus of historical German newspapers (801 pages), mainly from the period of the Austro-Prussian War (1866). The data contains detailed annotations of page layouts (paragraphs, titles, tables, images, separators) and more than 371,000 text lines with manually corrected transcriptions according to the OCR-D standard. A complete automated pipeline has been developed that contains the following steps:

- Layout segmentation (U-Net) – recognizes types of regions;
- Baseline detection (U-Net) – detects text lines;
- OCR (LSTM and Transformer models) – recognizes text.

In our paper also the need for a subsequent manual processing of the text is emphasized, together with the optimization of this process being our main focus.

Isaacs et al. (2024) presents the creation of three large humanitarian corpora in English, French, and Spanish, compiled from the ReliefWeb platform through its public API. The authors describe a methodology for language identification (since they use multilingual data), noise reduction, and automatic language processing (tokenization, lemmatization, and morphosyntactic annotation), as well as enrichment of the corpora with metadata. In our work, additional linguistic annotation, as well as metadata, will also be necessary. At this stage, however, we focus on the extraction of as accurate text as possible. We aim at ensuring the correct

order of the texts and preventing this order from blended newspaper articles. In this way, the results of any subsequent automatic language processing would improve significantly. Currently, we additionally annotate the metadata embedded in certain places in the print (mainly at the beginning and end) to facilitate their subsequent extraction.

Kjosbakken (2025) describes a practical approach to building a high-quality OCR pipeline, which is a key component in many machine-learning systems (e.g., document classification, RAG systems, receipt processing). The authors recommend testing different OCR systems and their subsequent evaluation. We use a similar approach to the definition of the algorithm settings, experimenting with different parameters and evaluating with a Levenshtein-type algorithm, which is one of the recommended ones by them.

In June (2026), the authors perform OCR using small language models and additional processing in the pipeline, which is also our goal. However, they are processing copy-heavy documents (forms, insurance, government documents, financial statements), which are highly structured and massively repetitive. This is the opposite of our type of documents, which have a very diverse structure and relatively limited repetition. This creates difficulty in implementing “rules” to ensure the quality of the text extracted from the model; therefore, more attention should be paid to post-editing and verification of the extracted text.

Gutehrlé and Atanassova (2021) presents a rule-based method for Logical Layout Analysis (LLA) of historical newspapers presented in an XML ALTO format. This paper also addresses the problem of text extraction from printed publications. The main focus is on the text type classification – especially the headline detection. The approach they used is to extract text formatting information (size, italic, bold) into a special xml format designed to encode such information – XML ALTO. At this stage, we have not determined the type of text pieces, but this is one of the necessary next steps. For us, the important lesson is that text formatting information, according to this paper, can improve the results.

Lee et al. (2020) presents the Newspaper Navigator Dataset, a collection of automatically extracted visual content from historical newspapers. The project uses more than 16.3 million digitized pages from the Chronicling America initiative (Library of Congress), covering the period 1789–1963. The authors develop a pipeline based on deep learning that automatically detects and extracts 7 types of visual content: headlines, photographs, illustrations, maps, comics, editorials, cartoons, advertisements. These findings do not fall directly into the line of our current work, since our focus is exclusively on text extraction. However, in our future work we plan to incorporate the available visual content.

3. Preparation of a Searchable PDF Dataset of Bulgarian Historical Newspapers

In this section, we present our approach towards providing a very high quality of the OCR result.

The newspapers chosen to participate in the dataset, in their original paper form, are stored in the "Ivan Vazov" National Library - Plovdiv. Considering the digitization process, the library fulfills two main tasks:

1. Preparation of high-quality "master" files for long-term archiving, which are typically 24-bit color scans at 300 ppi made directly from the original paper items. These files (usually in .tiff format) are not accessible to the general public.

2. Preparation of web-suitable PDF files of lower quality and size, acquired by running OCR with ABBYY FineReader software directly onto the master files. Then they are uploaded to the Digital Library to provide public access. The masters are compiled into a single PDF per cultural heritage unit, e.g. one PDF for one newspaper issue.

The most valuable public-domain material predates the 1945 orthographic reform. Therefore, the OCR quality is strongly affected by the historical alphabets and spelling conventions, the archaic vocabulary and mixed-language content, as well as the limited dictionary coverage. The age-related document issues (paper darkening, faded ink) and the uses of non-standard fonts inevitably reduce the recognition quality. A method for improving the accuracy of the OCR, provided by ABBYY FineReader, allows a targeted “training” for hard-to-recognize characters. It is particularly valuable for non-standard fonts and for historical Cyrillic characters as well as others that disappeared in the contemporary Bulgarian alphabet. For this task, we prepared a set of training snippet samples from different parts of the pages of the different newspaper issues. We also provided the corresponding machine-readable text within the clippings. Here are some examples in Fig.1:

Ideally, the training snippets must contain all the upper-case and lower-case letters of the alphabet, and all fonts to be found in a typographically stable period of a series of newspaper issues. The number of snippets used depends on the variety of fonts within the respective period. After the training based on the snippets is completed, all the trained symbols can be exported into the form of a *.fbtx*

Figure 1: Training snippets (on the left) and the recognized text (on the right).

file. This file can then be applied to all of the preselected newspaper issues. This is performed by loading the training file into the HotFolder software (part of the FineReader suite) for batch OCR processing. It should be noted that this procedure significantly improves the recognition quality, without resorting to post-OCR manual corrections.

Our plan is to process 309 periodical editions and about 150 000 pages. Each scanned issue is equipped with metadata consisting of: the newspaper title, the publisher, the date, and some other information. During the creation of the corpus, we have been trying to extend the coverage of the metadata given that we find additional information.

4. Extraction of Reading Orders over the Searchable PDF Dataset

In this section, we present our approaches towards the text extraction from Searchable PDF produced by the OCR software. The output of ABBYY FineReader can be stored in different formats. For our tasks, we consider two formats appropriate: *Searchable PDF* format and *MS Word docs* format. Both formats provide segmentation of texts into blocks that resemble the visual formatting of newspaper pages. In the work presented here, we are working with searchable PDF, because, as described in the previous section, there already exist 134 916 pages of scanned newspapers. It is not effective at all to perform the OCR of these newspapers again. At the same time, for the newspapers that have not been OCR-ed yet, we plan to store both formats.

Fig 2 shows a page from the “Belomorec” newspaper, in which the software segments, implemented by us, divide the text into many (relatively) small blocks. They need to be ordered in an appropriate way form a consistent and complete article. As mentioned above, in our current processing, we do not extract the images from the newspapers. This is left for future work.

Our task is after the initial segmentation of the content of the page in blocks to apply some heuristic algorithms to resize some of them and produce the correct blocks. In addition, it is necessary to determine the order of the blocks in such a way that the concatenation of their content forms the complete articles in the newspaper issue.

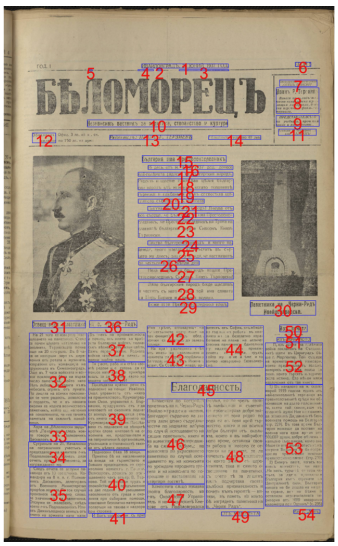

Figure 2: An example of a page where the blocks are ordered with the help of the reading order algorithm. The resulting order is consistent with both — the five-column layout and the partial horizontal separator.

The PDF documents for each newspaper issue in the dataset are presented in a searchable PDF. This means that within the PDF there is a layer of text aligned to the visual format of the paper. Each recognized character is connected to the coordinates that determine the place of the character occurrence. We are using the `pdfminer`[1] library to extract the text from PDFs. [2]

Such libraries use heuristic methods on the coordinates to compile the characters into lines and the lines into blocks. They also try to define a reading order of the blocks based solely on the coordinates. The order of the extracted blocks is often wrong - it fails to capture the layout of the newspapers and jumps between columns, making the text output

[1] https://github.com/euske/pdfminer

[2] We have performed experiments with several libraries for text extraction from searchable PDFs and after manual evaluation we selected `pdfminer`. After creating a gold corpus of 45 corrected and ordered newspapers, we compared the extracted texts from `pdfminer`, `pdfplumber`, `pymupdf`, and `pypdf` to the gold corpus using both normalized insertion and deletion ratio and Levenshtein distance. The scores are as follows: 0.8399 (486 714 edits), 0.4492 (1 520 982 edits), 0.8588 (408 803 edits) and 0.8504 (436 287). The scores are similar (with the exception of `pdfplumber`) and suggest that `pymupdf` is the best option according to these metrics.

incomprehensible. In this paper, we propose an algorithm that more successfully assigns the reading order of the blocks after extraction. We used the off-the-shelf order of the `pdfminer` blocks as a baseline for evaluation.

Our reading order algorithm works with blocks of text. Blocks are defined as a sequence of lines that form a coherent text. They are usually longer than one word, but shorter than an article, and have a rectangular shape in the document. We extracted the initial blocks with the `pdfminer` library. We view them determined by 4 numbers: (x_1, x_2, y_1, y_2) — the 2D coordinates of their bottom left corner — (x_1, y_1) and top right corner — (x_2, y_2). The `pdfminer` library uses coordinates, where (0, 0) is the bottom left corner of the page. Thus, in the representation of a block in this way always $x_1 \leq x_2$ and $y_1 \leq y_2$.

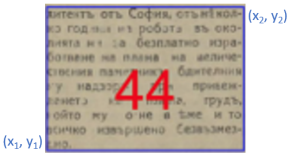

Figure 3: Block 44 from Fig. 2 represented by bottom left corner — (x_1, y_1) and top right corner — (x_2, y_2)

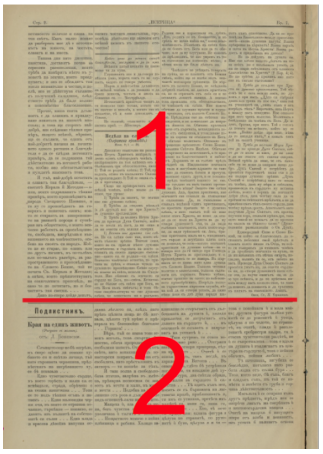

Figure 4: An example of a page segmented into two main areas, containing text from two articles. The areas are segmented horizontally. The red line is the gap between the two subpages, marked here with numbers 1 and 2, respectively.

As a first step, we find the *subpages* of the page. For this purpose, we find the horizontal gaps that expand from the left side to the right side of the page. They should not cross any blocks and should have at least some minimal thickness parameterized as *subpage_gap_threshold*. These gaps between blocks or subpages in the newspapers are marked with lines. However, the OCR software does not encode these lines in the text layer because they do not play any role in the search within the searchable PDF. Thus, we cannot use them for tracing the boundaries of the subpages or blocks. For that reason, we could only use the block coordinates to find such boundaries. The process of finding these gaps is as follows:

1. We initialize a list of candidate gaps from the bottom y coordinate of all blocks — y corresponds to y_1 for each block.

2. For each candidate, we test whether there is another block with a bottom y coordinate lower than the candidate gap and a top y coordinate higher than the candidate line plus a threshold of *subpage_gap_threshold* y coordinates. There is a block which is crossed by the candidate line and the candidate gap line cannot be a subpage boundary.

3. If the candidate gap does not cross a block, it becomes a gap that separates two subpages.

The gaps found in such a way separate the page in subpages and we consider them as boundaries of the subpapes — an illustration of such subpages is depicted in Fig. 4. The subpages are the first criterion for defining the reading order.

After identifying the subpages, our next goal is to find the subpage columns. The boundaries of the columns are recognized as vertical gaps:

1. With a step of *x_step* points we move horizontally from left to right. Each vertical line that spans at least *min_column_page_ratio* points of the page and does not cross a block (with a tolerance of *x_tolerance*), we mark as a candidate column separator.

2. If the column gap is larger, it is possible to get many duplicated separators close to one another. We remove each consecutive separator that is closer than 1.5x*x_step* points to the previous.

3. We also remove any separator that is closer than *min_column_width* points to the previous, thus defining the minimum column width.

4. Each separator defines a column to its right (the beginning of the page is also taken as a separator), and every block is assigned to the column of its closest left separator.

The columns are the second criterion for defining the reading order. Fig. 5 shows all correct separators for the columns.

Analogously to the vertical separators that define columns, there are also horizontal separators that change the reading order between the columns. Such segmentation of a column is depicted in Fig. 6.

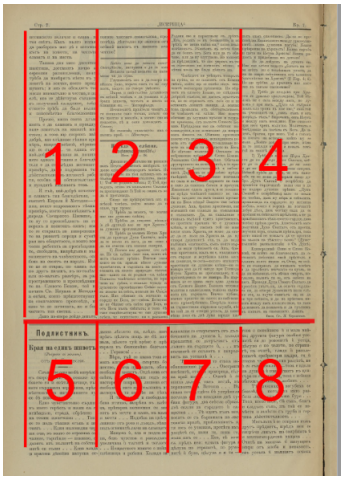

Figure 5: The same page from the Fig 4 additionally segmented vertically in columns which represent the order of the text in each horizontal area.

1. We find them in the same way as the full page horizontal separators by detecting the bottom y coordinates of blocks, which do not cross other blocks for every sublist of columns. For example, if we have 4 columns in the page, we try to find separators in the blocks of the sets of (1, 2, 3, 4), (1, 2, 3), (2, 3, 4), (1, 2), (2, 3) and (3, 4) columns. We define a minimal height of the gap as *partial_gap_threshold* points to account for the accidental occurrence of two adjacent blocks that end in adjacent points.

2. After finding these lines, we unify the adjacent lines with a tolerance of *y_tolerance* points.

3. We define the lines with a y coordinate and the minimum and maximum x coordinates of the blocks in their set of corresponding columns.

4. We also remove lines that are nested in longer lines.

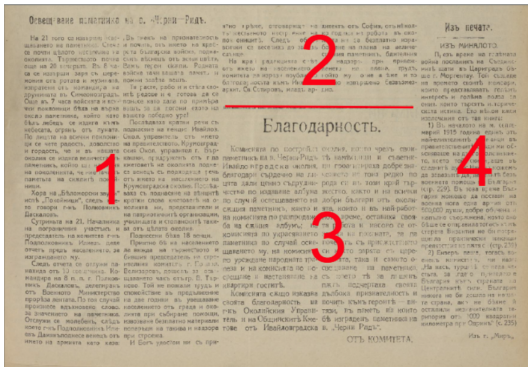

Figure 6: An example of a page with a partial horizontal separator that assigns a reading order to the blocks to the left, to the top, to the bottom, and to the right.

We sort the separators according to their coordinates from the beginning of the page — the bottom left corner of the page. Then, in the sorted order of the separators, each separator adds a new criterion to the final sorting of the blocks:

1. We assign 1 to all blocks left of the separator
2. 2 to all blocks higher than the separator
3. 3 to all blocks lower than the separator
4. 4 to all blocks right of the separator

After these steps, we employ a stable lexicographical sort on all blocks based on the aforementioned criteria. In other words, we sort by:

1. page number
2. subpage number
3. the numbers assigned by each of the partial horizontal separators
4. column number
5. top y coordinate
6. left x coordinate

These provide the reading order of the blocks on the page.

Problems can also arise. For example:

- Bad OCR or small font can make the algorithm skip a separator;
- A larger font can induce a false separator.

As mentioned above, the other output of the OCR is the *MS Word Docs* format. In this case, the ABBYY FineReader performs its own segmentation in blocks and groups them into larger blocks. These blocks are ordered according to the heuristics built in the software. Although this segmentation looks much better than the segmentation from PDFs, it suffers from the same problems: an incorrect union of smaller blocks which need to be reformatted, parasitic blocks within other blocks which need to rearrange; noise blocks. Fig. 7 depicts the two segmentations. We envisage in future to apply our algorithms to the *MS Word Docs* formats as well as to combine the two segmentations in order to exploit their mapping.

5. Evaluation

In this section, we evaluate the quality of the algorithm for determining the reading order of texts within the different blocks on the pages, as described in the previous section. The performance of the algorithm depends

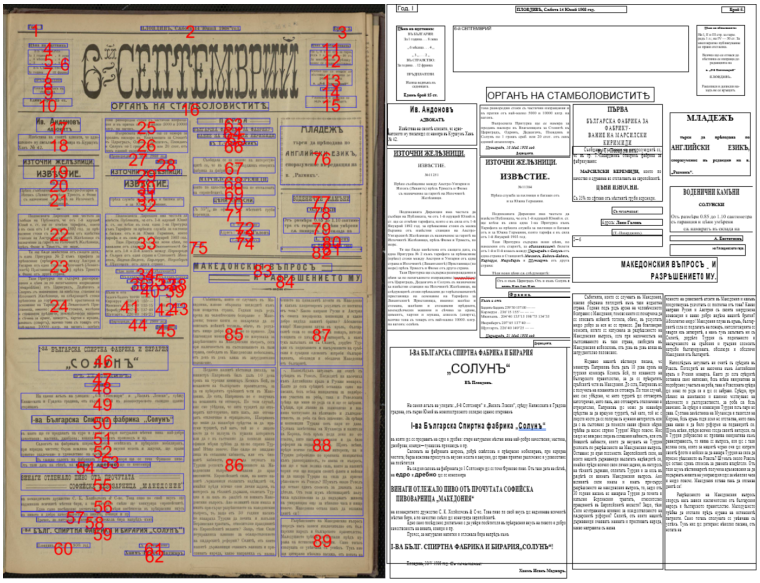

Figure 7: Here we present the segmentation of the PDF format and of the MS Word Docs format. As we could see the segmentation provided by the OCR software and stored in the MS Word Docs format is much more precise. Unfortunately it is not available to us for the already processed newspapers. We use them for the new scanned ones.

on its parameters — *subpage_gap_threshold*, *x_step*, *min_column_page_ratio*, *x_tolerance*, *min_column_width*, *partial_gap_threshold*, *y_tolerance*. The borderline cases of too large or too small values suggest automatic fine-tuning over the parameter values of the algorithm. In order to find the best parameters, we need a gold set of examples and a method of evaluation over the constructed reading order.

5.1. Our Setup

Some initial parameter settings were selected. Then we first processed, then manually checked and corrected the reading orders produced by the algorithm for 259 pages in 45 newspaper issues from different publishers with different typography. The corpus contains 1 878 295 characters, 379 395 tokens, and 10 251 blocks. In this way, we constructed a gold dataset for comparison among the reading orders produced with different parameter settings. As a baseline, we consider the reading order produced by the `pdfminer` library.

Once we have an available gold corpus of newspaper pages with correct reading order for each automatically generated reading order, we need to find a method to compare the two reading orders. For that purpose, we defined different operations over the reading orders such as *change scope of the block — dividing a block into two or more blocks; combining two or more blocks into one block*; change the order of the blocks. We formalized these operations as insert, delete, and replace of blocks in a reading order. When we had to compare two automatically generated reading orders, we decided to calculate the changing steps for each of these reading orders towards the gold standard reading order. Thus, we consider the reading order with the smallest number of steps to be better than the automatically generated reading order.

As a metric for the evaluation of the reading order, we defined a generalized version of the Levenshtein Edit Distance over blocks with respect to the changing operations mentioned above. The edit operations are the insertion and deletion of blocks, as well as the modification of block coordinates. Therefore, we consider two blocks as matching (or identical) if all of their coordinates are up to 5 points different. Using this metric, we can optimize the threshold parameters for all documents in the corpus. A possible approach is to optimize the threshold parameters only for a specific periodical edition, but this direction would require manual work to annotate at least one example.

5.2. Experimental results

We started the experiments for tuning the parameters by comparing both — the edit distance of the baseline ordering, produced by the `pdfminer`, and the ordering, produced by our algorithm with the manual set of parameters, to the ordering in the gold corpus. With initially manually set values of the parameters, we achieved a 33% reduction in the number of edits — from 2874 edits for the baseline, to 1925 edits for the initial version of the algorithm.

For fine-tuning the parameters, we performed

Baseline	Initial	Optimized
2874	1925	1702

Table 1: Edit distance between the manually annotated reading order and: the off-the-shelf library extraction (baseline) (**Baseline**), our page processing algorithm with initial intuitive set of parameters (**Initial**) and our page processing algorithm with optimized parameters over the gold dataset (**Optimized**).

Parameters	Initial	Optimized
x_step	5	5
x_tolerance	10	12
y_tolerance	20	20
subpage_gap_threshold	10	9
partial_gap_threshold	20	26
min_column_page_ratio	0.60	0.55
min_column_width	100	20
Edit distance	1925	1702

Table 2: Optimal values of the reading order parameters of the algorithm according to an extensive grid search over the gold dataset.

a grid search testing over 20,000 combinations of parameters of the algorithm, and compared the minimum edit distance between the manual annotated reading order of the gold set and the automatic reading order of the algorithm. This further reduced the edits to 1702. The test results are presented in Table 1. The optimal values for the parameters are given in Table 2.

Several experiments were also conducted with a subset of specific newspapers, where using certain individual parameter values, the results improved by up to 60% compared to the optimal values of the whole set. These results suggest that optimizing the parameter values to a specific subset of newspapers could be beneficial but requires some manual work.

The final result of the algorithmic extraction is an XML file with the following structure: pages → subpages → columns → blocks → text.

6. Manual Correction of the Extracted Reading Order

The algorithm for assigning the order of the blocks reduces manual work by a large margin. Still, there may be problems (some of them inherited from the block extraction of the `pdfminer` library):

- Noise blocks or whole columns;
- Blocks covering more than one column;
- Wrong block order.

Thus, if we want to produce an extraction of high quality, the manual correction is inevitable. At this stage of processing, our goal is the manual correction to be performed on the level of blocks. This would minimize the need to read all the internal text. Thus, we designed a software application in which users can edit the blocks themselves. The users can delete blocks and change their coordinates to have a better coverage over the text. The users can also change the reading order of the blocks. Our software visualizes changes in geometry and order of blocks immediately on the page. Furthermore, if the text is incorrectly divided into two blocks, it is easier to delete the second block and expand the first rather than merge the text.

Correcting the blocks has one more advantage, which can be beneficial in the long run. When we edit the block coordinates and do not move the text, we still have the mapping between the blocks and the characters inside them. In this way, we can use the correct block order to get the correct word reading order. In the future, one can train a model to produce a correct reading order of the words based on the 2D coordinates of the characters on the page.

For visualization and correction, we created an application that loads the XML file representing the blocks and the reading order as well as the corresponding PDF for presentation of the page with visualized blocks and their numbers — similar to the figures like Fig. 2. The application implements three commands for editing the blocks:

- Classification of blocks as normal, meta, and noise (classifying a block as noise is equivalent to deleting it). The meta and noise blocks will be skipped when extracting articles.
- Correction of the block coordinates — either a specific coordinate or all of them;
- Editing the block order by selecting the blocks to swap.

The application visualizes each change immediately. This is the approach used to create the gold dataset that was used to compare the algorithm performance. The practice shows that these three commands are sufficient to perform all the necessary edits.

7. Conclusion and Future Work

In this paper, we present an approach for defining a reading order over the blocks extracted from searchable PDFs of a Bulgarian Historical Newspaper Dataset. For this task we constructed a specialized gold dataset. We also propose a method that builds on a Levenshtein edit distance to measure the quality of the generated reading order and to

tune the parameters of the algorithm. As it can be seen, the approach is language independent.

Although the approach we presented here reduces the number of errors in the reading order, the amount of newspaper pages to be processed — over 150 000 pages in more than 300 titles, makes the manual inspection a tremendous task. Thus, we will proceed with the development of an automatic method to improve the processing of the data.

On the basis of the gold dataset, we will train several transformer-based language models. We plan to modify and fine-tune a pre-trained encoder transformer model to take as input subwords together with their 2D positions, and to output a text in the correct order.

Another future task is to add annotation of articles and metadata to the gold dataset in order to fine-tune the encoder article segmentation and classification models. In addition, spelling translation and correction will also be also performed with language models.

8. Acknowledgments

The reported work has been supported by CLaDA-BG, *the Bulgarian National Interdisciplinary Research e-Infrastructure for Resources and Technologies in favor of the Bulgarian Language and Cultural Heritage, part of the EU infrastructures CLARIN and DARIAH*.

9. Bibliographical References

Jonathan Bourne. 2025. Reading the unreadable: creating a dataset of 19th century english newspapers using image-to-text language models. *Digital Scholarship in the Humanities*, page fqaf151.

Xiaowen Ding and Weiqun Huang. 2014. Pdf converter production of historical newspaper digitization: The picture experience of china's dachenglaojiu database. In *Proceedings of the International Newspaper Conference, IFLA*. International Federation of Library Associations and Institutions (IFLA).

Nicolas Gutehrlé and Iana Atanassova. 2021. Logical layout analysis applied to historical newspapers. In *Proceedings of the Workshop on Natural Language Processing for Digital Humanities*, pages 85–94, NIT Silchar, India. NLP Association of India (NLPAI).

Loryn Isaacs, Santiago Chambó, and Pilar León-Araúz. 2024. Humanitarian corpora for English, French and Spanish. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8418–8426, Torino, Italia. ELRA and ICCL.

Florian June. 2026. Hybrid OCR-LLM: Not a bigger model, but a smarter pipeline. Online article. AI Exploration Journey (Substack / Medium).

Eivind Kjosbakken. 2025. How to develop a powerful ocr pipeline for machine-learning systems. Towards AI article.

Benjamin Charles Germain Lee, Jaime Mears, Eileen Jakeway, Meghan Ferriter, Chris Adams, Nathan Yarasavage, Deborah Thomas, Kate Zwaard, and Daniel S. Weld. 2020. The newspaper navigator dataset: Extracting headlines and visual content from 16 million historic newspaper pages in chronicling america. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 3055–3062, New York, NY, USA. Association for Computing Machinery.

Christian Schultze, Niklas Kerkfeld, Kara Kuebart, Princilia Weber, Moritz Wolter, and Felix Selgert. 2025. Chronicling germany: An annotated historical newspaper dataset.