Neural Computing and Applications (2022) 34:9593–9611
https://doi.org/10.1007/s00521-022-06948-5

ORIGINAL ARTICLE

Reading order detection on handwritten documents

Lorenzo Quirós[1] · Enrique Vidal[1]

Received: 15 April 2021 / Accepted: 10 January 2022 / Published online: 3 February 2022

Abstract
Recent advances in Handwritten Text Recognition and Document Layout Analysis have made it possible to convert digital images of manuscripts into electronic text. However, providing this text with the correct structure and context is still an open problem that needs to be solved to actually enable extracting the relevant information conveyed by the text. The most important structure needed for a set of text elements is their reading order. Most of the studies on the reading order problem are rule-based approaches and focus on printed documents. Much less attention has been paid so far to handwritten text documents, where the problem becomes particularly important—and challenging. In this work, we propose a new approach to automatically determine the reading order of text regions and lines in handwritten text documents. The task is approached as a sorting problem where the order-relation operator is automatically learned from examples. We experimentally demonstrate the effectiveness of our method on three different datasets at different hierarchical levels.

Keywords Document layout analysis · Reading order · Handwritten text recognition

1 Introduction

Automatic Text Recognition (ATR) systems, such as Optical Character Recognition (OCR), Handwritten Text Recognition (HTR) and Keyword Spotting (KWS), have received much attention in the past decades. Thanks to these advances, it is now possible convert digital images of handwritten text documents into electronic text and/or find relevant text elements in collections of these images.

Handwritten (or printed) document images are commonly analyzed through two main processes: Text Recognition [2, 4, 6, 13, 20, 22] and Document Layout Analysis (DLA) [1, 7, 18]. The first process deals with raw image-to-text conversion, typically at text-line level. The second process guides the first one to the parts of the document where text is present. It also helps to assemble the recognized text elements to provide a meaningful structure to the information conveyed by the text. Both process are complementary; for instance, a line-level transcript could be meaningless unless the lines are sorted in the correct order, or in the case of a table, the relationship between cells must be established in order to build a database from the transcripts. In these cases, the *reading order* of the text elements becomes essential to convert transcripts into useful information.

Of course, the reading order does not have to be unique or linear (e.g., a footnote can be read before a main paragraph, or after it, or we can stop in the middle of a paragraph, read a footnote and return to the main text). However, the reading order can be just assumed to be the most common way a user will sequentially read one text element after the other in order to gather the underlying information. Therefore, for the sake of simplicity, in this work, the reading order is assumed to be unique and linear.

Although DLA is a very broad and complex field, most works on DLA focus only on automatic page segmentation and classification, either at text-line level (e.g., baseline detection) [7, 15] or including region level segmentation [1, 18]. For these subproblems, satisfactory results are currently achieved in most cases. However, the reading order problem is generally overlooked and handled over to a heuristic mapping of the layout elements. A very important drawback of this idea is the large amount of specific domain knowledge required to deal with each

✉ Lorenzo Quirós
loquidia@prhlt.upv.es

Enrique Vidal
evidal@prhlt.upv.es

[1] PRHLT Research Center, Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

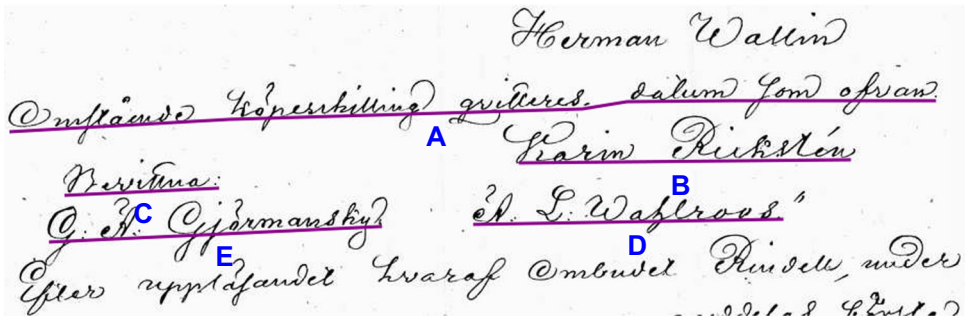


Fig. 1 Example of text lines where a naive top-to-bottom-left-to-right order would result in reading *A–B–C–D–E*, while the correct reading order is *A–C–E–B–D*

document collection. Moreover, most of the reading order approaches proposed so far are developed for *printed* documents.

Very little attention has been paid to handwritten text, where the problem is much more difficult. However, it is for handwritten documents in particular where a general solution to this problem is most needed.

For the sake of illustration, consider the small, yet complex, example shown in Fig. 1. In this example, a crop of a handwritten document with five text lines are depicted (represented by their baseline).

A naive order is to consider that a text-line s is *before* another text-line s', just if the vertical or horizontal coordinate of the gravity center of s is smaller than the one of s'; i.e., $y_s < y_{s'} \vee x_s < x_{s'}$. Such a simple, top-to-bottom-left-to-right (TBLR) order would lead to read *A–B–C–D–E*. However, the correct reading order in this example is *A–C–E–B–D*.[1]

TBLR is in fact the most traditional way to automatically determine the reading order of a document, and it is generally adequate for most *printed* text images and also for handwritten images with a simple and uniform layout (commonly after applying several geometrical corrections to the image). However, as illustrated in the simple example above, the correct, or desired reading order does not only depend on the (typically delicate) geometric positions of the text lines. In many handwritten text documents, like those shown in Figs. 3, 4, 5, 6, 7, 8, 9 the correct reading order also depends on the type of region each text line belongs to (paragraph, page-number, marginalia, etc.), its content and maybe other criteria which are heavily dependent on the types of documents considered. Clearly, in such cases, the naive approach fails.

Indeed, any automatic system that aims to obtain the reading order of such kind of documents should be able to handle the intrinsic uncertainly of the problem and it must be flexible enough to generalize to several variations in the documents of the same collection and variations between different collections. On the other hand, it worth noting that any manual attempt on the problem will require lots of human resources because most manuscript collections are huge.

In our work, we adopt a novel viewpoint for this problem: To automatically sort layout elements of handwritten documents into reading order, first, an order-relation operator is learned from examples, then this operator is used to decode the unknown reading order of new documents.

This viewpoint was introduced in [19], along with basic techniques to approach the decoding problem and preliminary experiments. Here, we extend our previous work through the following main contributions. First, we propose two new reading-order decoding algorithms with different computational costs. Then, we provide a theoretical background to these algorithms, as well as to the algorithm used in our previous work without formal deduction. Furthermore, we extend our previous experiments to the new decoding algorithms. Finally, we provide new experimental results for reading order extraction at different hierarchical levels, which proves to reduce computational costs and generally improves the quality of results.

The paper is organized as follows. Related work is discussed in Sect. 2. Section 3 describes the problem and the proposed approach. Afterward, experimental setup and results are presented in Sects. 4 and 5, respectively. Finally, in Sect. 6, we draw some conclusions and outline possible extensions for future work.

2 Relation with previous work on reading order

Research in the document reading order task has a relatively long tradition for *printed documents* [3, 11, 12, 14]. Details of how these works have influenced our proposal is detailed in [19]. In short, we follow the idea of assuming a pairwise partial order at element level from [3] and that using training data to automatically acquire the required domain knowledge from [12].

[1] The correct reading order is defined by an expert. Notice that, the lines *C*, *E* and *B*, *D* contain two-column tabular data (this happens frequently in this manuscript collection, where two or three groups of person names are listed columns-wise, while most of the other text lay in a single column across the page or in marginalia).

In contrast with printed text, only few works can be found in the literature that address the reading order problem for *handwritten documents*, where the layout does not generally or consistently follow a standard structure but typically just the art of the writer.

An interesting work is described in [16]. It focuses on table analysis, but it can be foreseen that the reading order could be obtained as a byproduct of the proposed process. The relations between table cells (or, in general, between any other type of layout elements) is modeled as a graph. The nodes represent layout elements (text lines only in their experimentation) and the edges represent the geometrical and logical relations between them. Conceptually speaking, an initial graph is built and then it is refined by means of a Graph Neural Network edge classifier, trained on examples of ground-truth graphs. While the graph representation is very attractive for this task, it suffers from the complexity associated with the analysis and computation of a fully connected graph. To render the problem tractable, they use a set of heuristic assumptions to reduce the number of initial edges. Clearly, these assumptions need to be revised and updated for each kind of document, which becomes a practical drawback of this approach.

In comparison with other techniques, the method we propose has the advantage of its simplicity and applicability to many kinds of documents without requiring specific domain knowledge. Equally important, the computational cost of most of the proposed decoding algorithms is reasonable for modern computers, as demonstrated here by experiments involving hundreds of elements per page.

3 Reading order through estimated binary order relations

Many handwritten text document have complex, heterogeneous and even erratic layout structures. To determine the reading order of this kind of documents, properties of the document beyond raw geometry are needed (as illustrated in Fig. 1). To circumvent this brittle and expensive expert knowledge dependency, we aim to learn the desired order directly from annotated training data. Notice that this training data is normally available as part of the ground-truth generated to train an automatic text recognition system.

The input of our proposed approach is not the document itself but a set of previously extracted layout elements (baselines, text regions, illustrations, etc.). Normally, a piece-wise linear curve is used to represent the baseline of a text line and a polygon to represent a region of interest.[2] In either case, several geometrical and typological features can be easily extracted from these representations, which can be used as input for the proposed statistical modeling. More details about these features are provided in Sect. 4.

3.1 Problem formulation

Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of n adequately represented layout elements of a page image. As discussed in Sec. 1, in this work, a given reading order in S is assumed to be unique and linear . Therefore, it can be defined as a *permutation*, usually denoted by a set $\mathbf{z}$ of n pairs $\{(s, v) : s \in S, v \in \mathbb{N}^{\leq n}\}$ where v, called "index," also satisfies $v_i \neq v_j, 1 \leq i, j \leq n \forall j \neq i$.

For instance, for the document in Fig. 1, the permutation $\mathbf{z}' = \{(A,1),(B,2),(C,3),(D,4),(E,5)\}$ represents a naive TBLR order in S, while the correct reading order is represented by $\hat{\mathbf{z}} = \{(A,1),(B,4),(C,2),(D,5),(E,3)\}$.

Our problem can be now be stated as follows: Given a set of layout elements S, obtain a permutation of S that renders its elements in reading order.

An order in S can alternatively be specified by means of a binary order relation "$\prec$" on $S \times S$ which fulfills the properties of a strict total order [5]. Thus, $\forall s, s' \in S$, $s \prec s'$ is assumed to mean that s is placed before s'. For instance, for the permutation $\hat{\mathbf{z}} = \{(A,1),(B,4),(C,2),(D,5),(E,3)\}$, we can state: $A \prec C$, $C \prec E$, $E \prec B$, $B \prec D$ and also,

$$A \prec E,\ A \prec B,\ A \prec D,\ C \prec B,\ C \prec D,\ E \prec D$$

. Hence, this binary order relation can be represented in matrix form as shown in this example:

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{1}$$

Note the rows and columns of this matrix can be arranged in many ways, leading to multiple matrix representations of the same permutation. Among these representations, there is one in which all the elements above the diagonal are 1 and those below the diagonal are 0:

$$\begin{array}{c} \\ A \\ C \\ E \\ B \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} A & C & E & B & D \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{2}$$

[2] Modern techniques to obtain these elements from raw handwritten text images can yield very accurate results and are fairly robust to image degradation and other typical difficulties of handwritten documents [1, 7, 15, 18].

This matrix is said to be in "canonical form" and obviously corresponds to the very same permutation $\hat{\mathbf{z}}$, which can also be written as: $\hat{\mathbf{z}} = \{(A,1),(C,2),(E,3),(B,4),(D,5)\}$. While assuming canonical representation might simplify notation in some cases, the ensuing formulation does not need such an assumption.

In general, any permutation $\mathbf{z}$ can be represented by a Boolean matrix $H^{\mathbf{z}} = [h^{\mathbf{z}}_{i,j}]^{n\times n}$, where each $h^{\mathbf{z}}_{i,j}$ is defined in terms of the indexes of $\mathbf{z}$ as:

$$h^{\mathbf{z}}_{i,j} = \begin{cases} 1 & \text{if } v_i \prec v_j\,, v_i, v_j \in \mathbf{z} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

If $\mathbf{z}$ represents a strict total order, $H^{\mathbf{z}}$ fulfills the following properties:

$$\begin{aligned} &h^{\mathbf{z}}_{i,i} = 0, 1 \le i \le n \\ &h^{\mathbf{z}}_{i,j} = 1 - h^{\mathbf{z}}_{j,i}, 1 \le i,j \le n,\ i \ne j \\ &h^{\mathbf{z}}_{i,j}. \end{aligned} \tag{4}$$

Conversely, for any matrix which represents a strict total order, the indexes defining the corresponding permutation $\mathbf{z}$ can be straightforwardly computed by just counting the number of zeros of each matrix row. We will capitalize on this observation later to propose a very fast decoding algorithm to determine the reading order using the results of learning $\prec$ from training examples.

Note, however, that not any arbitrary Boolean matrix represents a proper permutation; only if it fulfills the conditions (4), it can represent a strict total order.

Now, the approach we propose can be split into two main steps. First, a probabilistic order-relation operator is learned from samples (see Sect. 3.2). Then, the most probable reading order is decoded from the set of all probabilistic order relationships between all layout elements in the same page document (or hierarchical region), as given in detail in Sect. 3.3. A general diagram of this process is shown in Fig. 2.

3.2 Learning the order relation

The binary order relation $\prec$ defined in Sec. 3.1 can be explicitly rewritten as a function $\mathcal{R} : S \times S \to \{0,1\}$. Therefore, learning to sort layout elements into a correct reading order corresponds to learn $\mathcal{R}$ from training examples of its input pairs and binary outputs. To this end, training examples of correctly sorted layout elements are required.

Let $\mathbf{z} = \{(s_1, v_1), \ldots, (s_n, v_n)\}$ be the permutation corresponding to a correctly sorted set of layout elements of a

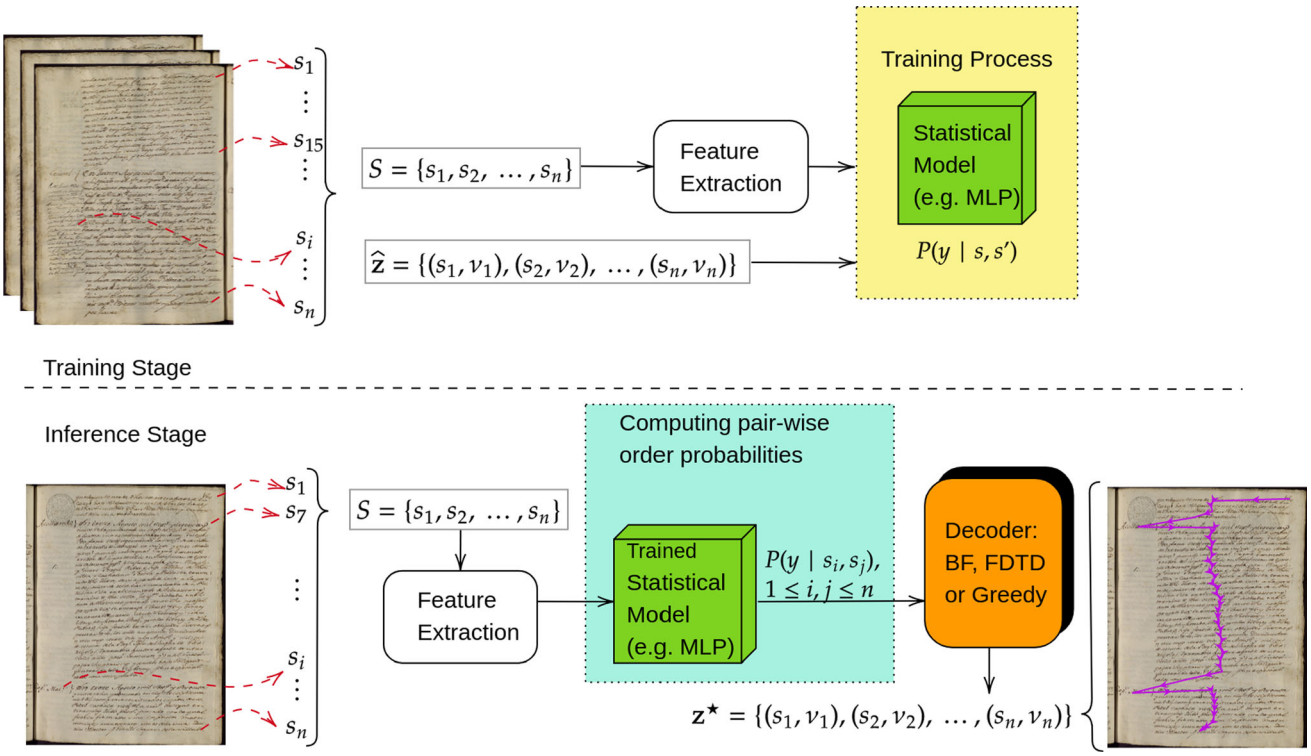


Fig. 2 General diagram of the proposed approach at training and inference or decoding stages

page image. From this permutation, we construct the training samples for $\mathcal{R}$ for all the possible pairs s_i, s_j as:

$$([s_i, s_j], y), y = \begin{cases} 1 & \text{if } v_i < v_j, 1 \leq i,j \leq n, \ i \neq j \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

A demonstration of how Eq. (5) is applied for the correct reading order of the example in Fig. 1 can be seen in [19].

This process is applied to the correctly sorted layout elements of all available training images, resulting in a training set which will be referred to as $\mathcal{D}$.

From $\mathcal{D}$, a model is trained to estimate the conditional distribution $P(Y = y \mid s, s')$, where Y is a Boolean random variable such that $Y = 1$ *iff* $s \prec s'$, and s, s' is the value of a random variable corresponding to an ordered pair of layout elements. Therefore, note that s and s' are not interchangeable in $P(y \mid s, s')$, that is, s, s' should *not* be seen as a conventional conjunction of two individual conditions but as a single condition by itself.

$P(y \mid s,s')$ can be estimated by means of any binary classifier. As discussed in Sect. 4.3, in this work, we estimate $P(y \mid s,s')$ using the well-known multilayer perceptron (MLP) method [21] for binary classification. To train the model, the layout elements of each training sample in $\mathcal{D}$ are represented as feature vectors composed of a few geometric and layout-related attributes. Details of the model used to estimate $P(y \mid s,s')$ is given in Sect. 4.

Due to the estimation process, it is possible that the estimated distribution fails to cover all the properties of the real distribution. Consequently, it may happen that this distribution lacks some properties which would be desired for a proper order-relation probabilistic model. For instance, it may come about that $P(Y = 1 \mid s,s') > 0.5$ (from which we may infer that $s \prec s'$), and also $P(Y = 0 \mid s',s) < 0.5$ (which suggests just the opposite, $s' \prec s$). This problem is most likely caused by the fact that $P(y \mid s,s')$ is only conditioned by two specific layout elements, ignoring the rest of the layout elements of the image. Ideally, for any pair $s_i, s_j \in S$, $P(Y = 1 \mid s_1, \ldots, s_i, s_j, \ldots, s_n)$ should be identical to $P(Y = 0 \mid s_1, \ldots, s_j, s_i, \ldots, s_n)$. In our experiments, we have rather seldom found that $P(Y = 1 \mid s,s')$ and $P(Y = 0 \mid s',s)$ are different. However, better overall performance can be achieved if we enforce strict equality by heuristically re-estimating the pairwise order probability as the average of $P(Y = 1 \mid s,s')$ and $P(Y = 0 \mid s',s)$:

$$\tilde{P}(y \mid s,s') = \frac{P(y \mid s,s') + 1 - P(y \mid s',s)}{2}, y \in \{0,1\}, \ s \neq s' \quad (6)$$

And, of course, $\tilde{P}(Y = 1 \mid s,s) = 0$, $\tilde{P}(Y = 0 \mid s,s) = 1$, $\forall s \in S$. Since $\tilde{P}(Y = 1 \mid s,s') + \tilde{P}(Y = 0 \mid s,s') = 1$ $\forall s, s' \in S, \tilde{P}(y \mid s,s')$ can still be properly interpreted probabilistically.

The values of $\tilde{P}(Y = 1 \mid s,s')$ can be arranged in matrix form, exactly as in the examples of Eq. (1) or Eq. (2). In fact, those matrices can be seen as the values of $P(Y = 1 \mid s,s')$, where probabilities only have the extreme values 0 or 1. A few illustrative examples of possible pairwise order-relation probability matrices for the five lines example of Fig. 1 are given in Appendix A.

3.3 Decoding a best reading order

Once the pairwise binary order-relation model is estimated, we discuss how to obtain the reading order of the set of layout elements of any new, unseen, text image. In general, the best reading order $\mathbf{z}^\star$ is a solution to the following optimization problem:

$$\mathbf{z}^\star = \arg\max_{\mathbf{z} \in Z} P(\mathbf{z}) = \arg\max_{\mathbf{z} \in Z} P(H^{\mathbf{z}}) \quad (7)$$

where a permutation $\mathbf{z}$ is described in terms of the matrix $H^{\mathbf{z}}$ (see Eq. (3)) and Z is the set of all possible permutations of S. We can express $P(H^{\mathbf{z}})$ as the joint probability of all its elements, and apply the chain rule, to obtain:

$$P(H^{\mathbf{z}}) = P(h^{\mathbf{z}}_{1,1}) P(h^{\mathbf{z}}_{1,2} \mid h^{\mathbf{z}}_{1,1}) \cdots P(h^{\mathbf{z}}_{n,n} \mid h^{\mathbf{z}}_{1,1}, \ldots, h^{\mathbf{z}}_{n,n-1}) \quad (8)$$

The values of the diagonal elements of $H^{\mathbf{z}}$ are deterministic (their value is 0 since they refer to the position of an element respect to itself). Hence, $P(h_{i,i} \mid \ldots) = 1$, $1 \leq i \leq n$. On the other hand, according to Eq. (3), the other elements of $H^{\mathbf{z}}$ satisfy $h_{i,j} = 1 - h_{j,i}, 1 \leq i,j \leq n$, $i \neq j$. Therefore, the conditional probability $P(h_{i,j} \mid \ldots)$ is completely defined for those elements conditioned by $h_{j,i}$, i.e., $P(h_{i,j} \mid \ldots, h_{j,i}, \ldots) = 1$. Consequently, $H^{\mathbf{z}}$ is completely defined by its strictly upper triangular part.

Using these properties of $H^{\mathbf{z}}$ a more compact version of Eq. (8) can be written. Moreover, assuming that the remaining elements of $H^{\mathbf{z}}$ are independent of each other[3], Eq. (8) can be written as:

$$P(H^{\mathbf{z}}) \approx \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} P(h^{\mathbf{z}}_{i,j}) \quad (9)$$

Now, using the pairwise order-relation estimates discussed in Sect. 3.2 (Eq. (6)), Eq. (9) can be expressed as:

$$P(H^{\mathbf{z}}) \approx \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} \tilde{P}(Y = h_{i,j} \mid s_i, s_j) \quad (10)$$

Numerical examples of this computation for the five lines example of Fig. 1 are provided in Appendix A.

[3] A stricter assumption were made in our previous work [19], where we assume independence of all elements in $H^{\mathbf{z}}$. In that case, the same optimization problem is defined, but the magnitude of the probability is scaled by a square term. We refer the reader to [19] for details.

Finally, form Eq. (7) and Eq. (10):

$$\mathbf{z}^{\star} \approx \arg\max_{\mathbf{z}\in Z} \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} \tilde{P}(Y = h_{i,j} \mid s_i,s_j) \tag{11}$$

The most straightforward decoding method is by exhaustive search or "brute force," that is, by checking all the $n!$ permutations of S. Clearly, this approach generally becomes intractable on handwritten documents, where a common page could range from few elements (e.g., $n = 20,\ n! = 2.4 \cdot 10^{18}$) to hundreds of elements (e.g., $n = 200,\ n! = 7.8 \cdot 10^{374}$). Nonetheless, in this work, we use this approach as reference when the number of elements is sufficiently small (see Sect. 5.2.2). Furthermore, numerical results of the Brute Force approach for the five lines example of Fig. 1 are provided in Appendix A.

In the following subsection, two efficient decoding methods are proposed to circumvent the huge complexity of exhaustive search.

3.3.1 Greedy decoding

The brute force approach guarantees a global optimal solution to Eq. (7), but it is computationally unfeasible for most real cases of interest. To overcome this impediment, we can resort to local optimization. A good approximation can be obtained by a greedy selection of the most probable layout element at each position $t, 1 \le t \le n$, using Algorithm 1.

Algorithm 1: Greedy algorithm to approximately decode the most probable reading order.

Data: a set of text lines, $S = \{s_1, \ldots, s_n\}$, and their pairwise probabilities, $\tilde{P}(Y = 1 \mid s,s'), \forall s,s' \in S$
Result: locally optimum reading order ($\tilde{\mathbf{z}}^{\star}$)

```
t=1
z̃* ← {∅}
while t ≤ n do
    b ← 0
    for s ∈ S do
        c ← ∏_{s'∈S, s≠s'} P̃(Y = 1 | s,s')
        if c > b then
            e ← s
            b ← c
        end
    end
    z̃* ← z̃* ∪ {(e,t)}
    S ← S \ {e}
    t++
end
```

By construction, Alg.1 ensures the resulting permutation, $\tilde{\mathbf{z}}^{\star}$, is *proper*. However, this greedy method is obviously sub-optimal and therefore the probability of $\tilde{\mathbf{z}}^{\star}$ is just a lower bound of the probability of the optimal permutation, that is,

$$P(\tilde{\mathbf{z}}^{\star}) \le P(\mathbf{z}^{\star}) \tag{12}$$

Nevertheless, according to our observations and results reported in Sect. 4, the bound is fairly tight. Therefore, if $\tilde{P}(Y = y \mid s,s')$ is well-estimated, good approximations to the global optimum are generally achieved.

Numerical results obtained with this method for the five lines example of Fig. 1 are provided in Appendix A.

3.3.2 First decide then decode (FDTD) decoding

Greedy decoding proves useful to overcome the computational complexity of searching for the global optimal solution. However, an even simpler and also more accurate solution can be achieved as explained here.

Following Eq. (10), an equation similar to Eq. (11) can be written without explicit permutation notation as follows:

$$H^{\star} \approx \arg\max_{H} \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} \tilde{P}(Y = h_{i,j} \mid s_i,s_j) \tag{13}$$

where $H = [h_{i,j} \in \{0,1\}]^{n\times n}$, with $h_{i,i} = 0, 1 \le i \le n$.

Notice that without any restriction of the relative values of the elements of H, a straightforward solution to the optimization problem (13) is achieved for a matrix such that each individual factor of the product is maximum. This allows us to solve Eq. (13) by simply setting for $1 \le i,j \le n$:

$$\begin{aligned} h^{\star}_{i,j} &= \arg\max_{y=\{0,1\}} \tilde{P}(Y = y \mid s_i,s_j) \\ &\equiv \begin{cases} 1 & \text{if } \tilde{P}(Y = 1 \mid s_i,s_j) > 0.5 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{14}$$

That is, instead of decoding the best solution directly from Eq. (13), we *first decide* which element in S is to be placed before the other. Then, as mentioned in Sect. 3.1, given $H^{\star}$, the indexes ($v^{\star}_i$) of the corresponding permutation can be obtained by just counting the number of zeros in the i-th row of H, that is:

$$v^{\star}_i = \sum_{j=1}^{n} (1 - h^{\star}_{i,j}), 1 \le i \le n \tag{15}$$

It is important to realize that while the matrix $H^{\star}$ computed according to Eq. (14) is an optimal solution to Eq. (13), the permutation extracted from $H^{\star}$ using Eq. (15) may be *improper*.

From Eq. (6), $\tilde{P}(y \mid s,s') = 1 - \tilde{P}(y \mid s',s)$ and for all s, $\tilde{P}(Y = 1 \mid s,s) = 0$. Therefore, from Eq.(15), $h^{\star}_{i,i} = 0$, $1 \le i \le n$ and $h^{\star}_{i,j} = 1 - h^{\star}_{j,i}$, $1 \le i,j \le n$, $i \ne j$. So the two first conditions in (4) required for $H^{\star}$ to represent a proper permutation are fulfilled. Yet, $H^{\star}$ may still fail to satisfy the last condition in (4). For instance, since $\tilde{P}(Y = y \mid s_i,s_j)$

does not take into account the complete context of S, the following cases can arise:

- $\tilde{P}(Y = 1 \mid s_i, s_j) > 0.5$,
- $\tilde{P}(Y = 1 \mid s_j, s_k) > 0.5$,
- $\tilde{P}(Y = 1 \mid s_k, s_i) > 0.5$.

Consequently, Eq. (14) leads to obtain $h^{\star}_{i,j} = 1$, $h^{\star}_{j,k} = 1$ and $h^{\star}_{k,i} = 1$, which contradicts the last condition in (4) (i.e., given the first two values, $h^{\star}_{k,i}$ is expected to be 0). As a result, we obtain ties in the number of zeros per row in $H^{\star}$, and Eq. (15) yields repeated values of elements of $v^{\star}$.

Clearly, if $H^{\star}$ does not have ties, the corresponding permutation is an optimal solution to Eq. (11). Otherwise, note that the optimization problem (13) is equivalent to a relaxed version of the problem (11) where one of the restrictions has been dropped. Therefore, the probability of Eq. (10) computed for $H^{\star}$ is an upper bound of the maximum probability associated with the optimal $\mathbf{z}^{\star}$ of Eq. (11), that is,

$$P(H^{\star}) \geq P(H^{\mathbf{z}^{\star}}) \tag{16}$$

Moreover, according to our observations and results reported in Sec. 4, the bound is fairly tight. This suggests that even in the cases where $H^{\star}$ does not correspond to a proper permutation, we can still obtain a close to optimal (proper) permutation by adequately breaking the ties in the number of zeros per row in $H^{\star}$. While several heuristics can be used for this purpose, in the present work, we just solve the few ties observed arbitrarily by assigning the first element in the detected tie to the position v_i under analysis and the other element to the next position.

This approach is illustrated in Appendix A for the five lines example of Fig. 1.

3.4 Hierarchical approach

Although the proposed model has been described at the page level, it can be also used within any smaller layout element previously detected in the page. In general, this leads to a hierarchical approach, where Eq. (6) and any suitable decoder is applied in each hierarchical level independently.

Nevertheless, beware that any error in a higher level will place all its sub-elements in the wrong position, increasing the risk of obtaining an highly incorrect reading order. However, a hierarchical approach can also simplify a process of manual correction, since a single fix in a given level may lead to correctly sort all the corresponding sub-elements in the lower level.

4 Experiments

In order to evaluate the applicability and capability of the proposed methods to solve the reading order problem, we consider three datasets with different degrees of complexity and training data availability.

In contrast with our previous work [19], where the reading order problem was only considered for text-lines at the page image level, here we obtain the reading order for three tasks associated with basic hierarchical layout levels and elements, namely text lines at page level, regions at page level and text lines at region level. More specifically, we will present experiments corresponding to a) sorting the raw text lines of a page image (ignoring possible text regions of the page, as in [19]), b) sorting the text regions of a page image and c) sorting the text lines within each text region. In addition, we also report results, comparable to some extent with those of task a), of a hierarchical combination of tasks b) and c).

Along with the proposed approach we use the top-bottom-left-right approach (TBLR) as a basic benchmark in all the experiments. This simple technique has been chosen because it is perhaps the only one known so far which can be consistently and uniformly applied to the several types of page layouts exhibited by the handwritten images used in the experiments here presented. Other more complex methods depend on very specific domain knowledge and heuristics that should be updated for each kind of document, which prevents a homogeneous application to various types of layouts.

As commented in Sec. 3, the input data for each of these tasks is not the page image itself, but a set layout elements previously determined through layout analysis. For each element we use a set of features extracted from its layout analysis description, including both geometric and categorical attributes. Specifically:

- Text line (defined by its baseline piece-wise linear curve):

  Region type the text line belongs to, one-hot encoded
  Normalized coordinates of the center of the baseline
  Normalized coordinates of the leftmost baseline end
  Normalized coordinates of the rightmost baseline end

- Text Region (defined by its bounding polygon):

  Region type
  Normalized area of the polygon
  Normalized center of mass of the polygon
  Extreme left, right, top and bottom normalized coordinates of the polygon.

Note that in all experiments, we assume that the layout elements are previously revised and amended if necessary. Therefore, the input can be assumed to be ground-truth.[4]

Note also that the features described above do not include any information about the text possibly included in each layout element (line). An explanation for this choice is in order.

Clearly, as stated in Sec. 1, in many cases, it can be difficult or impossible to tell a correct reading order without taking into account the content of the layout elements (the transcript in the case of text lines).

However, there are also many other cases where textual content is not an adequate cue to determine reading order. For example, the cells in a *table* should be ordered according to the table structure, as given by column and/or raw headers, and the raw cell contents provides little or no information about which cell is to be considered before another. Similarly, the order of fields in many structured documents such as *forms* can often be better estimated more from visual than from textual features. In these cases, visual layout features become more important than the content itself.

Therefore, estimating the reading order should be seen as a twofold problem, where both visual and textual cues can be important or not depending on the task. The results presented in the next section show that using only simple visual or geometric features can already provide enough accuracy to allow the use of the proposed approaches in many real applications. And for the most difficult types of layouts, such as tables, it is unclear that textual features might actually help to improve the results.

Nevertheless, of course, our results do still leave significant room for improvement, and we do think that in many cases it might actually come from the use of textual features which, in future works we plan extract using a recent methodology known as *"probabilistic indexing"* [17, 23].

4.1 Datasets

We consider three datasets with complex reading order: "Oficio de Hipotecas de Girona" (OHG),[5] "Finnish Court Records" (FCR)[6] and "READ ABP Table dataset" (ABP).[7]

Table 1 summarizes the main features of these datasets and the corresponding splits into training, validation and test parts. Details are provided in the following subsections.

4.1.1 Oficio de Hipotecas de Girona

This collection, composed of hundreds of thousands of notarial deeds from 1768–1862, is provided by the Centre de Recerca d'Història Rural from Universitat de Girona(CRHR).[8] Sales, redemption of censuses, inheritance and matrimonial chapters are among the most common documentary typologies in this collection. Digital page images are accompanied by their respective ground-truth layout in PAGE-XML format, compiled by the Handwritten Text Recognition group of the Pattern Recognition and Human Language Technologies[9] center and CRHR. OHG pages exhibit a relatively complex layout composed of six region types, namely *pag*, *tip*, *par*, *pac*, *not*, *nop*. An example is shown in Fig. 3.

A main complexity of this dataset is the diversity of region types and the fact that the number of regions in a page image varies widely (from just a single *pac* region, to up to nine regions of diverse types).

In this work, we use a publicly available portion of 596 pages from the collection, with the same data splits defined in [19] which are also summarized in Table 1. More details are provided in Table 3 of Appendix B. Splits are available online along with the source code used in the present experiments (see Sect. 4.4).

4.1.2 Finnish court records

The FCR dataset is a selection of 500 pages from the Renovated District Court Records (19th century),[10] a large collection of the National Archives of Finland. The documents consist of records of deeds, mortgages, traditional life-annuity, among others.

This dataset contains images with one or two document pages,[11] annotated with text lines and six different region types, namely *pageNum*, *marg*, *textRegion*, *textRegion2*, *table* and *table2*. A double-page example is shown in Fig. 4. The blend of single- and double-page images is a common complexity added to the DLA problem and the reading order problem itself. We select this dataset not only because of that complexity but also by the number of different regions as well.

[4] This assumption allows us to use standard metrics to assess the obtained results in a clearly understandable manner. Obviously, the proposed methods can straightforwardly be applied to automatically extracted (unsupervised) layout elements, but assessing the results with respect to unaligned references would make the evaluation protocol more obscure.

[5] https://zenodo.org/record/1322666.

[6] https://zenodo.org/record/3945088.

[7] https://zenodo.org/record/1243098.

[8] http://www2.udg.edu/tabid/11296/Default.aspx.

[9] https://prhlt.upv.es.

[10] https://arkisto.fi/en/records-3/copy-of-kansallisarkiston-aineistot.

[11] Nevertheless, the word "page" will be used in the rest of this paper to refer to each single or double-page image of a collection.

Table 1 Main characteristics of the datasets

Dataset	OHG	FCR	ABP
Pages (total)	596	500	111
Training	149	125	28
Validation	149	125	28
Test	298	250	55
Average lines per page	40	64	268
Average regions per page	4.9	3.8	7.7
Average lines per region	8.2	16.8	34.8

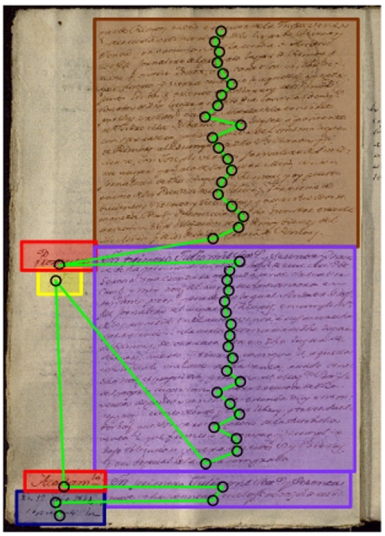


Fig. 3 An example of the OHG dataset. The text line reading order is depicted in green over the baseline centers, and text regions are depicted as *pac* in brown, *tip* in red, *par* in violet, *nop* in yellow and *not* in blue. Better seen in color

In this work, we use the same data splits defined in [19] which are also summarized in Table 1. More details are provided in Table 4 of Appendix B. Splits are available online along with the source code used in these experiments (see Sect. 4.4).

4.1.3 READ ABP table

It is a subset of the ABP_S_1847-1878 dataset, which contains information about the parishioners who died within the geographic boundaries of the various parishes of the Diocese of Passau between the years 1847 and 1878. The ABP dataset contains a very heterogeneous set of pages where the main element is a table. It is composed of 111 manually annotated pages which amount to 29 752 text lines or 15 231 cells. Only two different regions are defined, namely: *textRegion* for normal text regions and *tableRegion* for tables.

Since no reading order was explicitly defined in the dataset, we defined it as follows (see an example in Fig. 5):

1. *textRegion* elements are sorted in TBLR order
2. Then, *tableRegion* cells are ordered row by row from top-to-bottom
3. Finally, text lines in each cell are sorted in TBLR order

In this work, we use the same data splits defined in [19] which are also summarized in Table 1. More details are provided in Table 5 of Appendix B. On average, each ABP page contains 7.7 regions and 268 text lines. The number of lines is so large because each cell in a table may contain several small text lines. Splits are available online along with the source code to replicate the defined reading order and the experiments (see Sect. 4.4).

4.2 Metrics

Evaluating the reading order is a challenging task that requires measuring not only how many elements are placed in the correct position within the order, but also how far those elements are paced with respect to the correct position. Hence, basic metrics such as precision–recall or the number of misplaced elements are not enough, as they do not take into account the distance between the predicted placement of an element and the correct position.

Therefore, we resort to metrics used in information retrieval specifically for this purpose, such as Spearman's footrule distance and the Kendall's Tau rank distance. Moreover, as those metrics will take into account how many elements are misplaced and how far the elements are from the correct position, they give us and idea not only of the performance of the proposed methods but also an estimation of the effort needed to manually correct any error, as will be explained in the following paragraphs.

Normalized Spearman's footrule distance [10] is the normalized cumulative sum of distances between pairs in two ordering indexes, computed as:

$$\rho(\boldsymbol{t}, \boldsymbol{v}) = \frac{\sum_{i=1}^{n} \mid t_i - v_i \mid}{\lfloor \frac{1}{2} n^2 \rfloor} \tag{17}$$

where $\boldsymbol{t}, \boldsymbol{v}$ are the ground-truth and the hypothesis ordering indexes for S, $\lfloor \frac{1}{2} n^2 \rfloor$ is the maximum cumulative distance possible between all pairs (t_i, v_i), and $0 \leq \rho(\cdot) \leq 1$.

The normalized Spearman's footrule distance gives us the insight into not only how many elements are misplaced, but also how far are those elements from their correct position.

Kendall's Tau rank distance [8] is a metric also called bubble-sort distance, since it is equivalent to the number of swaps the bubble-sort algorithm would need to transform the order defined by $\boldsymbol{v}$ into the reference order defined by $\boldsymbol{t}$.

Hence, it can be interpreted as an upper-bound to the number of edit steps an expert user will need to make in order to edit the hypothesis into the correct reading order. Formally, it is defined as the number of discordant pairs between $\boldsymbol{t}$ and $\boldsymbol{v}$:

$$K(\boldsymbol{t},\boldsymbol{v}) = |\{(i,j) : i<j \wedge ((t_i<t_j \wedge v_i>v_j) \vee (t_i>t_j \wedge v_i<v_j))\}| \quad (18)$$

Hierarchical evaluation, in a hierarchical approach, the previously discussed metrics can be straightforwardly used to assess the results obtained at each level of the hierarchy. The resulting values will help to understand the effectiveness of the method at each level individually.

Furthermore, to compare a hierarchical approach to its flat full-page counterpart, the results obtained for the hierarchical ordering can be flattened into an order of the lowest hierarchy elements (lines in our experiments) at the full-page level. Then, the normalized Spearman's footrule distance can be directly computed as in the non-hierarchical case.

On the other hand, since any editing step (swap) in a higher level will update its sub-elements in a lower level as well, the Kendall's Tau rank distance should be simply computed as the sum of the swaps needed at each hierarchical level.

4.3 Experimental setup

One of the key points of the proposed method is to obtain a good estimate of $P(y \mid s,s')$ —and $\tilde{P}(y \mid s,s')$, Eq. (6).

In a previous work [19]we estimated this probability using different classifiers, and we found the Multilayer Perceptron (MLP) [21] model to be the most adequate for the problem. Consequently, we have also adopted this model in the present work, whose parameters are selected using the validation set of each dataset: three layers, with the number of neurons in the hidden layer set to twice the number of input features for each dataset; ReLU activation function in the input and hidden layers and sigmoid in the output layer. The ADAM optimizer [9] has been used for training, with a learning rate of 0.001.

Notice that the memory demands during training could be huge since the number of pairs generated per page is $n^2 - n$, where n is the number of layout elements. In order to reduce the memory footprint, instead of using all possible pairs of the training set, a random set of pairs is generated on the fly directly from the training samples (layout elements) in each epoch. The pairs are composed of the i-th layout element in the training set and another element selected randomly from the same page or region. In this way all the training samples are visited in each epoch, but the memory required is linear with the number of samples.

We perform three main types of experiments. First, the proposed decoding methods are compared. Second, the reading order of text lines at page level is extracted. Third, we split the process into two hierarchical reading orders, i.e., we sort the regions of each page, then the lines in each region are sorted too. Finally, we flatten the two hierarchical results into a text line order at page level.

To allow easy comparisons, we report the macro-average of each metric, relative to the number of test pages or

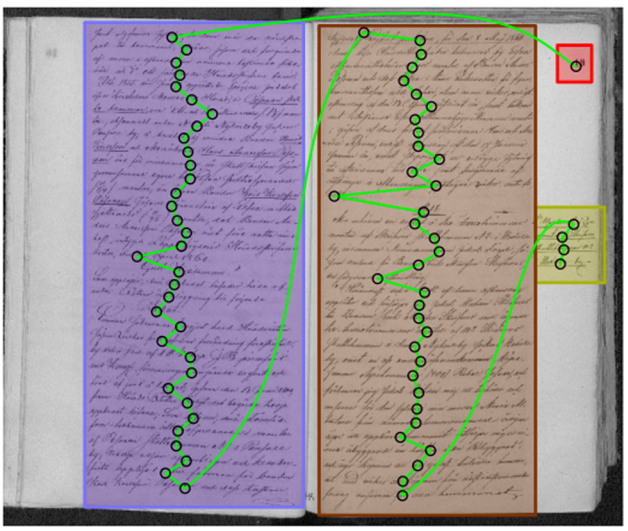


Fig. 4 A double-page example of the FCR dataset. The text line reading order is depicted in green over the baseline centers, and regions are depicted as *pageNum* in red, *marg* in yellow, *textRegion* in violet and *textRegion2* in brown

Fig. 5 An example of the ABP dataset. The text line reading order is depicted in green over the baseline centers, and the *textRegions* and *tableRegion* in red and violet, respectively

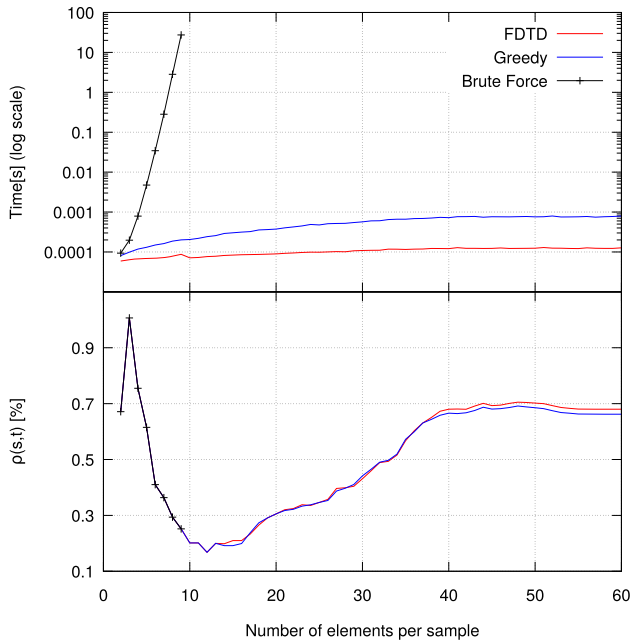


Fig. 6 Computing time and effectiveness ($\rho(s,t)$) of each decoding method for increasing input sizes. Better in color

regions of interest of each dataset. In addition, to reduce variance, all the values have been obtained as the average over ten experiments with different random initialization of the model parameters. Furthermore, $\rho(\boldsymbol{s},\boldsymbol{t})$ values are reported as a percentage (%) to avoid very small numbers.

4.4 Reproducibility

The lack of available datasets and tools is one of the main bottlenecks toward a reproducible research. For this reason, the source code used to carried out the experiments presented in this paper is freely available[12], along with the subsets used as training, validation and test for each dataset[13]. Similarly, the images and ground-truth files of all the datasets are available through the channels specified in Sect. 4.1.

We hope open-source code and datasets will help to boost research on document reading order and related areas.

5 Results

First, in Sect. 5.1, we evaluate how the size of the input sample influence the behavior of the proposed decoding algorithms. Then, in Sect. 5.2, we analyze the performance of the proposed method in different datasets.

5.1 Analysis of the decoding algorithms

In this experiment, we compare the three different decoding algorithms, presented in Sect. 3.3, for an increasing number of input layout elements. To this end, we build subsets of m text lines per page, from $m = 2$ to the maximum number of lines available (i.e., $2 \leq m \leq n$). The text lines were selected randomly from the OHG dataset (where $n = 40$ on average).

We compute the time each algorithm needs to obtain the hypothesis for m increasing from 2 to 60. In each case, the Spearman's footrule distance is computed as well in order to analyze the relative effectiveness of each decoding method. The results are summarized in Fig. 6.

As expected, the Brute Force computing time grows exponentially and only results for $m \leq 9$ could be obtained under a reasonable amount of time. By comparison, the computing times of both proposed decoders grow slowly, the Greedy method typically taking an order of magnitude longer than the FDTD decoder.

With respect to the accuracy of each decoder, for $m \leq 9$ all of them provide the same (optimal) response. Then after $m = 15$, the results are very similar for Greedy and FDTD, but toward $m > 38$ the latter starts behaving slightly better.

In the light of these results, the FDTD method exhibits better overall performance, taking into account both complexity and accuracy. Nevertheless, for the sake of completeness, in the following experiments, we report results for both the Greedy and the FDTD decoders (and also for the Brute Force, whenever possible).

5.2 Performance results

Results for the different datasets, tasks, decoding methods and metrics considered are reported in Table 2. Detailed discussion on these results follows in the coming subsections.

In general, both proposed decoding methods, Greedy and FDTD, achieve very similar performance in all datasets, tasks and metrics. Also, both methods significantly outperform the trivial TBLR approach, in most cases dramatically.

[12] https://github.com/lquirosd/Order_Relation_Operator

[13] https://github.com/lquirosd/Order_Relation_Operator/tree/master/data.

Table 2 Layout element ordering results for different metrics, tasks and decoders

Metric		$\rho(\boldsymbol{t},\boldsymbol{v})(\%)$				$K(\boldsymbol{t},\boldsymbol{v})$			
Dataset	Decoder	LP	RP	LR	LHP	LP	RP	LR	LHP
OHG	TBLR	2.91	9.34	0.06	3.51	12.971	0.614	0.012	0.67
	Greedy	0.70	0.22	0.05	0.06	3.449	0.019	0.010	0.07
	FDTD	0.69	0.21	0.05	0.06	3.400	0.020	0.011	0.07
	BF	–	0.20	–	–	–	0.017	–	–
FCR	TBLR	31.84	28.43	0.65	31.38	606.972	1.860	1.859	8.32
	Greedy	1.28	1.73	0.39	1.07	22.472	1.158	1.179	4.24
	FDTD	0.94	1.79	0.39	1.11	16.113	1.144	1.185	4.28
ABP	TBLR	10.93	16.59	0.92	8.77	3953.000	5.054	219.805	1707.54
	Greedy	7.97	6.26	0.60	5.03	2968.230	1.530	111.220	863.64
	FDTD	7.83	6.30	0.60	5.06	2936.150	1.589	111.305	963.04

Reported figures are page or region averages of values of $\rho(t,v)$ (in %) and $K(t,v)$ (absolute numbers of swaps). Tasks correspond to: ordering Lines at Page level (LP), Regions at Page level (RP), Lines at Region level (LR) and Lines obtained through Hierarchical processing via RP, but evaluated at Page level (LHP). The decoders are: Top Bottom Left Right (TBLR), Greedy, First Decide Then Decode (FDTD) and Brute Force (BF). Order-relation probabilities are learned using a multilayer perceptron. Each result is the average over ten randomly initialized experiments. In both metrics, the lower the better.

5.2.1 Text lines at page level

In this task, abbreviated as LP in Table 2, we obtain the reading order of all text lines present in each page for all three datasets. In general, both proposed decoding methods (Greedy and FDTD) perform very similarly in all datasets.

Results obtained for the OHG dataset are very encouraging. For the $\rho(\boldsymbol{t},\boldsymbol{v})$ metric, values below 0.7% are reported for both proposed decoders. This means that even though a few order errors exist, the text lines involved are placed near the correct relative position.

Also, given that each page in the dataset contains an average of 40 text lines, the expected effort to fix any error is very low, according to the $K(\boldsymbol{t},\boldsymbol{v})$ metric, on average less than 4 swaps per page would be needed by a user to achieve the correct reading order. As compared with the approximately 13 swaps required by TBLR, the improvement according to this metric is 74%.

Two representative examples of real OHG results are shown in Fig. 7. In the first one (left), the result provided by the proposed decoder is fully correct. This is achieved in spite of complexities like lines that are very close to each other at the top and the middle of the image. In the second example (right), the proposed method produces only one error, when a text line belonging to a side note (*nop*) is misplaced in the middle of a paragraph. In comparison, TBLR results are much worse, particularly in the second example.

It is important to notice that even the single error in the reading order of the second example inserts a text from a text region into another region. Such an apparently minor error may dramatically change the meaning of the corresponding paragraphs. This is a clear indicator of the difficulty and sensitivity of the reading order problem.

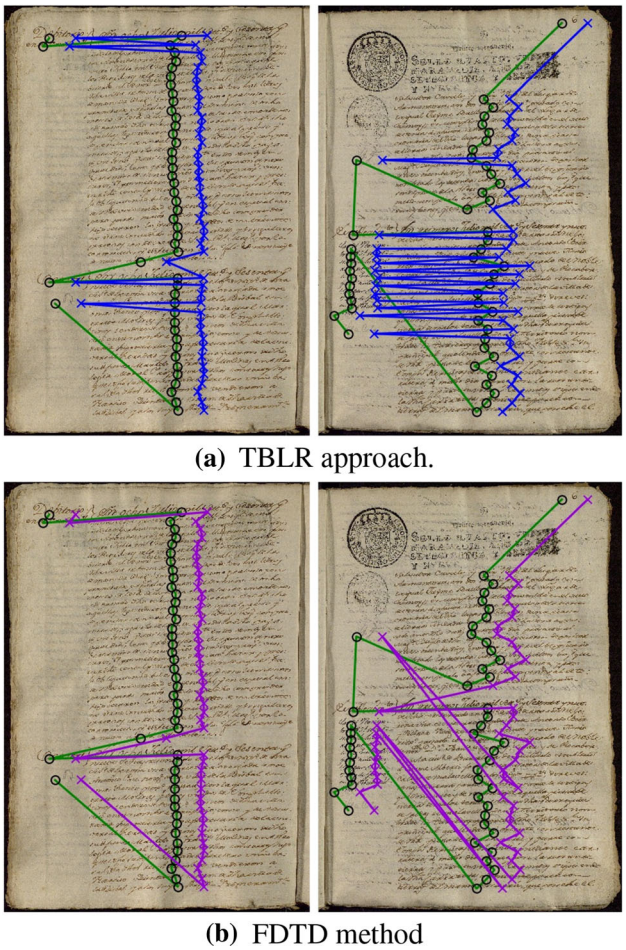
The FCR dataset has 64 text lines per image on average, with the complexity that an image can contain one or two document pages. A good quality reading order is also obtained by the proposed methods for this dataset. Values of $\rho(\boldsymbol{t},\boldsymbol{v})$ fall below 1% of the maximum displacement the elements can have. And an average of 16.1 swaps would be needed to correct the reading order, which is 97% lower than with the TBLR approach.

The largest performance differences between the two proposed decoders are observed in this dataset, with FDTD performing better than Greedy by 0.34% in the $\rho(\boldsymbol{t},\boldsymbol{v})$ metric and by more than 6 swaps per page according to $K(\boldsymbol{t},\boldsymbol{v})$.

Two representative examples of results are shown in Fig. 8. FDTD provides great results (only minor errors in the marginalia regions) In contrast, the results of the simple TBLR approach are hardly usable, particularly (as expected), those of the double-page example.

In contrast with OHG and FCR, the ABP dataset is very heterogeneous and only a few samples of each type of table are available for training. Moreover, it has as many as 268 average text lines per page. Figure 9 shows two examples of results for images that have a sufficiently small number of elements to allow visualization and readability.

Despite the difficulties, the proposed methods are able to obtain reasonably good qualitative results between rows in the same table, while the intra-row results are far insufficient for a useful reading order. Moreover, the average number of swaps needed to fix the errors is very large.



(a) TBLR approach.

(b) FDTD method

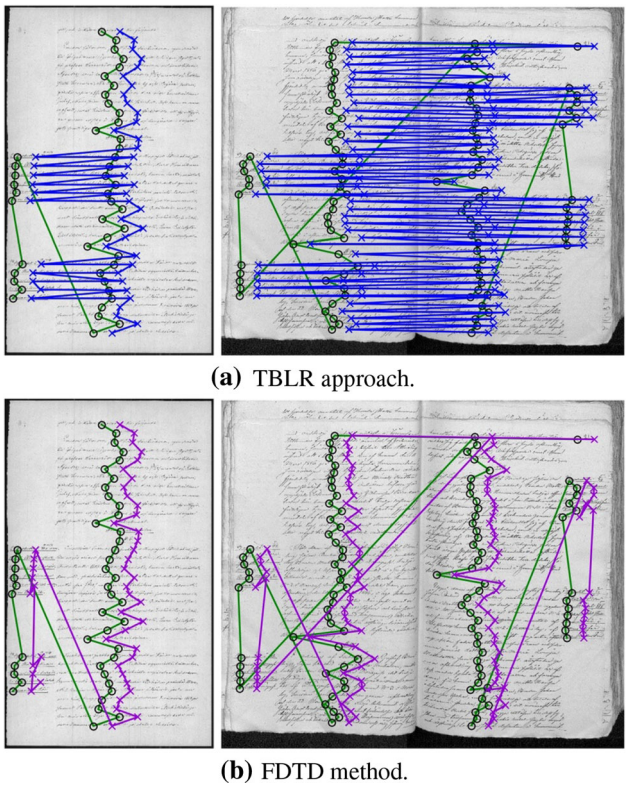
Fig. 7 Two examples (left-right) of results obtained for the OHG dataset where ground-truth is depicted in green and system results in blue (for the TBLR approach) and violet (for FDTD). The center of each element is slightly shifted to improve readability. Better seen in color

Nevertheless, the proposed methods still perform around 47% better than TBLR according to the $\rho(\boldsymbol{t}, \boldsymbol{v})$ metric.

The problems with this dataset are most probably caused by the scarce training samples along with the heterogeneous data in the dataset.

5.2.2 Hierarchical reading order

In these experiments all the regions of interest in the document are sorted according to the reading order, then the text lines inside each region are sorted as well. We consider each of these two tasks individually, followed by a hierarchical composition of the individual results at page level. Notice that this experiments, in contraposition to the plain approach (Sect. 5.2.1), requires a DLA system to previously segment the documents into regions of interest.



(a) TBLR approach.

(b) FDTD method.

Fig. 8 Two examples (left–right) of results obtained for the FCR dataset where ground-truth is depicted in green and system results in blue (for the TBLR approach) and violet (for FDTD). The center of each element is slightly shifted to improve readability. Better seen in color

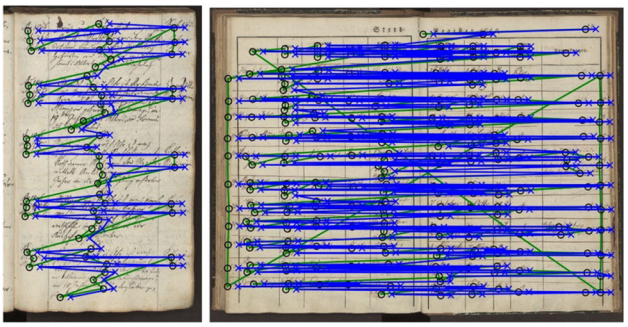
5.2.3 Region reading order at page level

In this task, abbreviated as RP in Table 2, we obtain the reading order of the text regions of each document page.

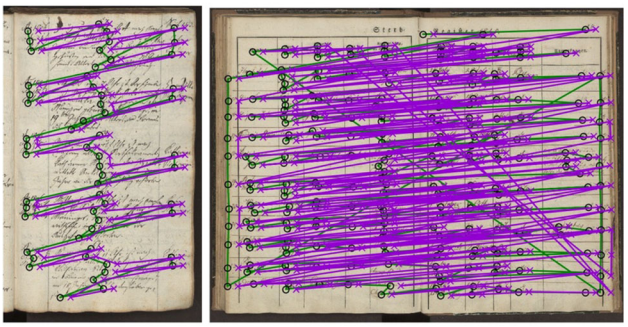
The number of regions of interest in any page of the OHG dataset is sufficiently small to allow using the Brute Force method. This way all the decoders can be empirically compared using reasonable computing resources.

As in the case of text lines at page level, in this experiment, all the proposed decoding methods perform similarly well and significantly better than TBLR. The very small difference between the Brute Force method (which guarantees optimal solutions) and the other two proposed methods is particularly worth noting. As already observed in the results reported in Sect. 5.1, this proves that the prohibitive optimal solutions are very well-approximated by the inexpensive proposed methods.

Using these methods, the average number of required region swaps ($K(\boldsymbol{t}, \boldsymbol{v})$ / RP, in Table 2) is less than 1 every 5 pages for OHG, about 1 per page for FCR and less than 2 per page for ABP. Equally important is to notice that $\rho(\boldsymbol{t}, \boldsymbol{v})$ is very small for OHG and FCR, which means that the very few misplaced elements are very near the correct positions.



(a) TBLR approach.



(b) FDTD method.

Fig. 9 Two examples (left-right) of results obtained for the ABP dataset where ground-truth is depicted in green and system results in blue (for the TBLR approach) and violet (for FDTD). The center of each element is slightly shifted to improve readability. Better seen in color

5.2.4 Text lines reading order at region level

In this task, abbreviated as LR in Table 2, text lines within each text region are sorted in reading order. Accordingly, the results presented in the LR columns of Table 2 are region averages instead of page averages as in the other columns.

In the OHG dataset, writing inside the text regions is very consistent. For this reason, even the results provided by the simple TBLR approach are reasonably good. Yet, the proposed methods still perform slightly better.

Likewise, in the FCR dataset the text line reading order produced by TBLR at region level is fairly good, but the proposed methods achieve 40% better results for $\rho(\boldsymbol{t}, \boldsymbol{v})$ and 36% for $K(\boldsymbol{t}, \boldsymbol{v})$ (number of swaps).

With respect to the ABP dataset, the proposed methods are able to reduce the average number of swaps to less than 112 (which is about half of those required by TBLR). However, the number of misplaced elements is still exceedingly large to be useful for real applications.

5.2.5 Hierarchical consolidation

In this task, abbreviated as LHP in Table 2, text lines are ordered through hierarchical processing. First text regions are sorted at page level, then text lines within each region are sorted. As discussed before, the performance of this task is assessed by evaluating the flattened text line order at page level. This allows us to compare this approach with the task of directly sorting text lines at page level (LP in Table 2).

In general, the hierarchical approach significantly overcomes the LP results, in some cases by more than one order of magnitude. For instance, in OHG, $K(\boldsymbol{t}, \boldsymbol{v})$ is reduced from 3.4 swaps to 0.07 swaps, using either the FDTD decoder or the Greedy approach. Important reductions are also achieved for FCR (from more than 16 to 4 swaps) and for ABP (from more than 2935 to 864 swaps). Regarding the $\rho(\boldsymbol{t}, \boldsymbol{v})$ metric, results also improve dramatically in the OHG dataset (from 0.7% down to 0.06%) and less so for FCR.

The simple TBLR results are also improved, but the proposed methods still outperform TBLR by large margins.

In summary, the hierarchical treatment of the data proves to be very important both to reduce the problem complexity and to increase the effectiveness of the proposed methods. Although these improvements come at the price of requiring a richer layout analysis where the text regions are accurately recognized, while in the LP task (text lines at page level), only plain text line detection is strictly required.

Examples of hierarchical results are shown in Figs. 10, 11 ,12 for the same pages for which the direct LP results are shown in Figs. 7, 8, 9.

In the OHG examples, both TBLR and the proposed FDTD method are able to obtain the correct reading order within the text regions, but only FDTD achieves the correct region reading order as well. As compared with the results of direct line ordering at page level, hierarchically using FDTD allows fixing the error shown in the right example of Fig. 7.

In the FCR examples, single page samples are well handled by both TBLR and FDTD, but for complex samples, such as double-page, including several marginalia, the simple TBLR approach still falls short, while the proposed method does achieve perfect results. As compared with direct FDTD line ordering at page level, using FDTD hierarchically allows fixing the subtle errors shown in Fig. 8.

In the ABP examples, finally, results for both methods are still so messy that probably will be easier for a human to establish the reading order from scratch than to fix errors.

6 Conclusions

We propose a new general approach to extract the reading order of handwritten documents based on learning a pairwise relation operator. Also, two different decoding

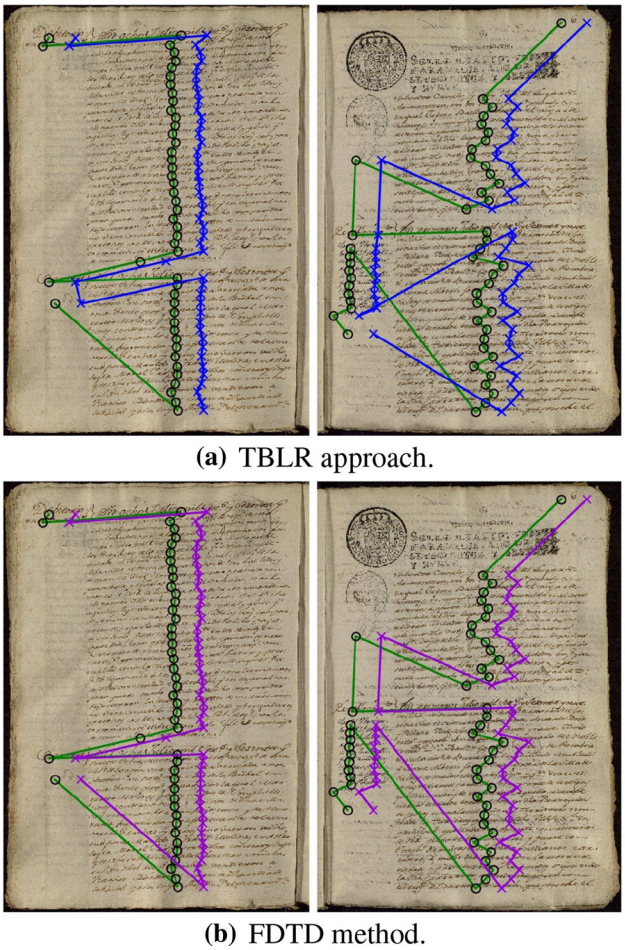


Fig. 10 Two examples (left-right) of hierarchical results obtained in the OHG dataset. The ground-truth is depicted in green and the hypothesis in blue for the TBLR approach, and violet for FDTD. The center of each element is slightly shifted to improve readability. Better seen in color

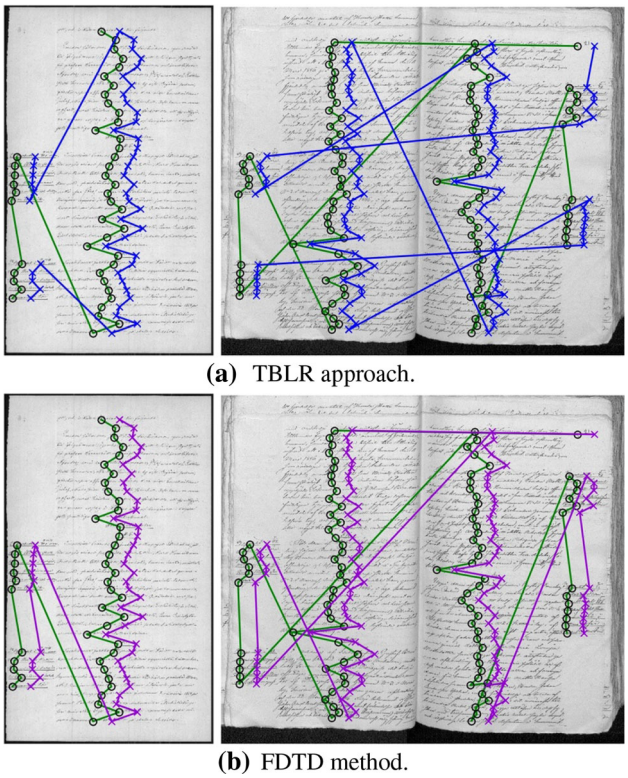


Fig. 11 Two examples (left-right) of hierarchical results obtained in the FCR dataset. The ground-truth is depicted in green and the hypothesis in blue for the TBLR approach, and violet for FDTD. The center of each element is slightly shifted to improve readability. Better seen in color

methods are presented, which provide much better ordering accuracy than a classical approach (TBLR) which sorts layout elements according to their geometric positions in the image. In general, the accuracy of the two proposed decoders is very similar, while the main difference is the computational complexity of each one. Experiments support that the FDTD algorithm is faster and slightly more accurate.

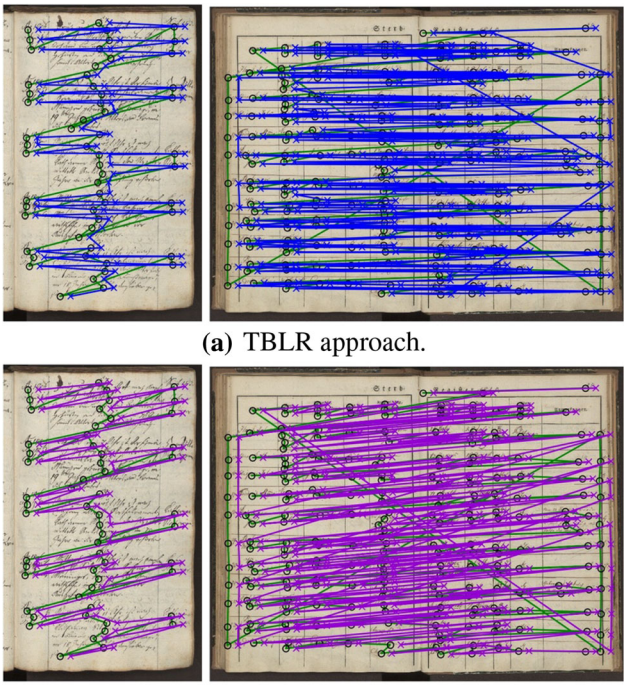
We report very good results in moderately homogeneous datasets such as OHG and FCR, while results in the very heterogeneous ABP dataset are still far away from being useful to recover the information present in the documents.

Furthermore, we study a hierarchical application of the proposed methods and show that such a hierarchically processing further reduces the complexity of the problem and increases the accuracy of the results.

Nevertheless, the method still exhibits some limitations that should be taken into account in any production scenario. In particular, the method depends on a good estimation of $P(y \mid s, s')$, which is proven to be hard in very heterogeneous datasets. Also, the computational cost of samples with a large number of elements (e.g., maritime navigation charts with thousands of elements) should be carefully analyzed. Moreover, errors in automatically detected layout elements (e.g., merged elements, mislabeled elements) could directly impact the proposed method.

In the future, we would like to address the problem of heterogeneous datasets by exploring more powerful classifiers, as well as more complex features to represent layout elements, including textual content features obtained by means of probabilistic indexing [17, 23]. Equally important to this end is to extend the proposed methods to take into account a more complete context of each layout element. We expect this will lead to more robust ordering models.

Finally, we aim at further exploring the algorithmics of the order decoding problem. According to Eq. (12), the probability of a solution obtained by the proposed Greedy method is a *lower bound* of the probability of a globally optimal solution. On the other hand, according to Eq. (16), the probability of a solution yield by the FDTD method is



(a) TBLR approach.

(b) FDTD method.

Fig. 12 Two examples (left-right) of hierarchical results obtained in the ABP dataset. The ground-truth is depicted in green and the hypothesis in blue for the TBLR approach and violet for FDTD. The center of each element is slightly shifted to improve readability. Better seen in color

an *upper bound* of the optimal probability. Moreover, as discussed before, both bounds are fairly tight. These results pave the way for the development of Branch and Bound methods that provide globally optimal solutions, as the Brute Force method considered in this work does, but requiring only moderate computation resources.

A Numerical examples

This appendix covers illustrative examples of possible learned values of $\tilde{P}(Y=1 \mid s_i,s_j), 1 \leq i,j \leq n$ for the five lines example in Fig.1 and how Eq. (10) is computed using these values. These examples also aim to help to understand important aspects of the decoding methods proposed in Sect. 3.3 and the effect of some corner cases on these methods.

Case 1

Here, we present an example of the most common case, where $P(Y=1 \mid s_i,s_j)$ is well-estimated:

$$\tilde{P}(Y=1 \mid s_i,s_j): \begin{array}{c} \\ A \\ C \\ E \\ B \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} A & C & E & B & D \end{array} \\ \begin{pmatrix} 0 & 0.6 & 0.9 & 0.8 & 0.7 \\ 0.4 & 0 & 0.8 & 0.6 & 0.9 \\ 0.1 & 0.2 & 0 & 0.7 & 0.9 \\ 0.2 & 0.4 & 0.3 & 0 & 0.8 \\ 0.3 & 0.1 & 0.1 & 0.2 & 0 \end{pmatrix} \end{array} \tag{19}$$

We can now use these values to apply Eq. (10) to the matrix in Eq. (2), corresponding to the canonical form of the permutation $\hat{\mathbf{z}} = \{(A,1),(B,4),(C,2),(D,5),(E,3)\}$:

$$\begin{aligned} P(H^{\hat{\mathbf{z}}}) \approx & \tilde{P}(Y=1 \mid A,C)\tilde{P}(Y=1 \mid A,E) \\ & \tilde{P}(Y=1 \mid A,B)\tilde{P}(Y=1 \mid A,D) \\ & \tilde{P}(Y=1 \mid C,E)\tilde{P}(Y=1 \mid C,B) \\ & \tilde{P}(Y=1 \mid C,D)\tilde{P}(Y=1 \mid E,B) \\ & \tilde{P}(Y=1 \mid E,D)\tilde{P}(Y=1 \mid B,D) \\ = & 0.6 \cdot 0.9 \cdot 0.8 \cdot 0.7 \cdot \\ & 0.8 \cdot 0.6 \cdot 0.9 \cdot \\ & 0.7 \cdot 0.9 \\ & 0.8 \\ \approx & 0.0658 \end{aligned}$$

The same result is obtained for the matrix in Eq. (1), but using the corresponding values of $\tilde{P}(Y=h_{i,j} \mid s_i,s_j)$ as:

$$\begin{aligned} P(H^{\hat{\mathbf{z}}}) \approx & 0.8 \cdot 0.6 \cdot 0.7 \cdot 0.9 \cdot \\ & (1-0.4) \cdot 0.8 \cdot (1-0.3) \cdot \\ & 0.9 \cdot 0.8 \cdot \\ & (1-0.1) \\ \approx & 0.0658 \end{aligned}$$

And now for the naive TBLR order $\mathbf{z}' = \{(A,1),(B,2),(C,3),(D,4),(E,5)\}$:

$$\begin{aligned} P(H^{\mathbf{z}'}) \approx & 0.8 \cdot 0.6 \cdot 0.7 \cdot 0.9 \cdot \\ & 0.4 \cdot 0.8 \cdot 0.3 \cdot \\ & 0.9 \cdot 0.8 \cdot \\ & 0.1 \\ \approx & 0.00209 \end{aligned}$$

The Brute Force decoding goes over the 120 different permutations to obtain that the most probable permutation is $\mathbf{z}^{\star} = \hat{\mathbf{z}}$, with $P(H^{\mathbf{z}^{\star}}) = 0.0658$. The Greedy and FDTD decoders also predict the same permutation.

Case 2

Another possible case, where $\tilde{P}(Y=1 \mid s_i,s_j)$ is properly estimated (in the sense discussed in Sect. 3.3), could be:

$$\tilde{P}(Y=1 \mid s_i,s_j): \begin{array}{c} \\ A \\ C \\ E \\ B \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} A & C & E & B & D \end{array} \\ \begin{pmatrix} 0 & 0.6 & 0.8 & 0.8 & 0.7 \\ 0.4 & 0 & 0.9 & 0.9 & 0.9 \\ 0.2 & 0.1 & 0 & 0.7 & 0.7 \\ 0.2 & 0.1 & 0.3 & 0 & 0.9 \\ 0.3 & 0.1 & 0.3 & 0.1 & 0 \end{pmatrix} \end{array} \tag{20}$$

Following Alg.1, the most probable element to be placed in the first position is C with $b = 0.2916$ (see lines $6-11$ in Alg.1). Upon termination, Alg.1 finally predicts the reading order as $\mathbf{z}^{\star} = \{(C,1), (A,2), (E,3), (B,4), (D,5)\}$ with $P(H^{\mathbf{z}^{\star}}) = 0.0576$ (which in fact is the second best permutation), while the global optimum permutation is $\mathbf{z}^{\star} = \{(A,1), (C,2), (E,3), (B,4), (D,5)\}$ with a higher probability, $P(H^{\mathbf{z}^{\star}}) = 0.0864$. The same optimal results is also obtained by the FDTD decoder in this case.

This is a representative example of the limitations of the Greedy decoder, where a local maximum corresponds to a sub-optimal result.

Case 3

As discussed in Sect. 3.3.2, it is possible that Eq. (14) leads to ties in the number of zeros per row in $H^{\star}$, resulting in an improper permutation. This may happen, for instance, in the following case, where $\tilde{P}(Y=1 \mid B,C) > 0.5$, $\tilde{P}(Y=1 \mid C,E) > 0.5$ and $\tilde{P}(Y=1 \mid E,B) > 0.5$:

$$\tilde{P}(Y=1 \mid s_i,s_j): \begin{array}{c} \\ A \\ C \\ E \\ B \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} A & C & E & B & D \end{array} \\ \begin{pmatrix} 0 & 0.6 & 0.9 & 0.8 & 0.7 \\ 0.4 & 0 & 0.8 & 0.4 & 0.9 \\ 0.1 & 0.2 & 0 & 0.7 & 0.9 \\ 0.2 & 0.6 & 0.3 & 0 & 0.8 \\ 0.3 & 0.1 & 0.1 & 0.2 & 0 \end{pmatrix} \end{array} \tag{21}$$

This combination contradicts the transitivity property of a total order and, using Eq. (14) and Eq. (15), FDTD yields an *improper* permutation, namely: $\mathbf{z} = \{(A,1), (C,3), (E,3), (B,3), (D,5)\}$, with $P(H^{\star}) = 0.0658$ obtained using Eq. (10). The optimal solution provided by Brute Force (and in this case also by the Greedy method) is $\mathbf{z}^{\star} = \{(A,1), (C,2), (E,3), (B,4), (D,5)\}$, with $P(H^{\mathbf{z}^{\star}}) = 0.0438 < 0.0658 = P(H^{\star})$. Depending on how the ties for C, E, B are resolved, different *proper* permutations can be obtained. Only if they are resolved in favor of $C \prec E \prec B$, the optimal permutation, $\mathbf{z}^{\star}$, is obtained.

B Details of the datasets

See Tables 3, 4 and 5.

Table 3 Main characteristics of the OHG dataset, which is divided into 149 pages for training, 149 for validation and 298 for test

Region Name	#Regions				#Lines			
	Train	Val	Test	Total	Train	Val	Test	Total
par	228	210	431	869	4051	3721	7597	15369
pac	111	123	228	462	1501	1815	3300	6616
tip	224	206	424	854	240	218	458	916
pag	79	77	135	291	79	77	135	231
nop	97	112	192	401	102	112	189	403
not	10	7	17	34	70	26	101	197

On average, each page contains 4.9 regions and 40 text lines

Table 4 Main characteristics of the FCR dataset, which is divided into 125 pages for training, 125 for validation and 250 for test

Region Name	#Regions				#Lines			
	Train	Val	Test	Total	Train	Val	Test	Total
pageNum	86	87	168	341	92	87	169	348
marg	132	142	308	582	637	729	1659	3025
textRegion	183	195	381	714	4731	4707	9326	18764
textRegion2	59	70	129	258	2216	2525	4740	9481
table	2	5	12	19	55	95	261	411
table2	1	2	4	7	9	52	87	148

On average, each image contains 3.8 regions and 64 text lines

Table 5 Main characteristics of the ABP dataset, which is divided into 28 pages fro training, 28 for validation and 55 for test

Region Name	#Regions				#Lines			
	Train	Val	Test	Total	Train	Val	Test	Total
textRegion	190	182	370	742	213	205	424	842
tableRegion	28	28	58	114	6956	7636	14318	28910

On average, each image contains 7.7 regions and 268 text lines

Acknowledgements The authors want to thank to the Centre de Recerca d'Història Rural, the National Archives of Finland and Déjean Hervé for facilitating the datasets used in this work, and to Juan Miguel Vilar for the enlighment comments. Also, this work was partially supported by Universitat Politècnica de València under grant FPI-II/900, by Generalitat Valenciana under the EU-FEDER Comunitat Valenciana 2014-2020 grant IDIFEDER/2018/025 "Sistemas de fabricación inteligente para la indústria 4.0", by the Ministerio de Ciencia, Innovación y Universidades project DocTIUM (Ref. RTI2018-095645-B-C22), by the BBVA Foundation through the 2019 Digital Humanities research grant "HistWeather – Dos Siglos de Datos Climáticos" and by the Agencia Estatal de Investigación under project SimancasSearch (PID2020-116813RB-I00).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Ares Oliveira S, Seguin B, Kaplan F (2018) dhSegment: A generic deep-learning approach for document segmentation. CoRR **abs/1804.10371**
2. Bluche T (2015) Deep neural networks for large vocabulary handwritten text recognition. Ph.D. thesis, Ecole Doctorale Informatique de Paris-Sud - Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur . Discipline : informatique
3. Breuel TM (2003) High performance document layout analysis. In: 2003 Symposium on document image understanding (SDIUT'03)
4. Coquenet D, Soullard Y, Chatelain C, Paquet T (2019) Have convolutions already made recurrence obsolete for unconstrained handwritten text recognition? In: 2019 International conference on document analysis and recognition workshops (ICDARW), pp. 65–70. IEEE, Sydney, Australia
5. Davey BA, Priestley HA (1990) Introduction to lattices and order. Cambridge University Press, Cambridge
6. Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2009) A novel connectionist system for unconstrained handwriting recognition. IEEE Transact Pattern Anal Mach Intell 31(5):855–868
7. Grüning T, Leifert G, Strauß T, Labahn R (2018) A Two-Stage Method for Text Line Detection in Historical Documents. CoRR **abs/1802.03345**
8. Kendall MG (1938) A new measure of rank correlation. Biometrika 30(1/2):81–93
9. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. 3rd International Conference on Learning Representations (ICLR)
10. Kumar R, Vassilvitskii S (2010) Generalized distances between rankings. pp. 571–580. https://doi.org/10.1145/1772690.1772749
11. Lee JY, Park JS, Byun H, Moon J, Lee SW (2002) Automatic generation of structured hyperdocuments from document images. Pattern Recognition 35(2):485–503
12. Malerba D, Ceci M, Berardi M (2008) Machine learning for reading order detection in document image understanding. In: Machine learning in document analysis and recognition, pp. 45–69. Springer, Berlin and Heidelberg
13. Martínek J, Lenc L, Král P (2020) Building an efficient OCR system for historical documents with little training data. Neural Comput Applicat 32(23):17209–17227
14. Naoum A, Nothman J, Curran J (2019) Article segmentation in digitised newspapers with a 2d markov model. In: 2019 International conference on document analysis and recognition (ICDAR), pp. 1007–1014
15. Pastor M (2019) Text baseline detection, a single page trained system. Pattern Recognit 94:149–161
16. Prasad A, Déjean H, Meunier J (2019) Versatile layout understanding via conjugate graph. In: 2019 International conference on document analysis and recognition (ICDAR), pp. 287–294
17. Puigcerver J (2018) A probabilistic formulation of keyword spotting. Ph.D. thesis, Univ. Politècnica de València
18. Quirós L (2018) Multi-task handwritten document layout analysis. CoRR **abs/1806.08852**
19. Quirós L, Vidal E (2021) Learning to sort handwritten text lines in reading order through estimated binary order relations. In: 2020 25th International conference on pattern recognition (ICPR). In press
20. Romero V, Serrano N, Toselli AH, Sánchez JAn, Vidal E (2011) Handwritten text recognition for historical documents. In: Proc. of the Workshop on language technologies for digital humanities and cultural heritage, pp. 90–96. Hissar, Bulgaria
21. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
22. Sánchez JA, Romero V, Toselli AH, Villegas M, Vidal E (2019) A set of benchmarks for handwritten text recognition on historical documents. Pattern Recognit 94:122–134

23. Toselli AH, Vidal E, Romero V, Frinken V (2016) HMM word graph based keyword spotting in handwritten document images. Information Sci 370–371:497–518