Noname manuscript No.
(will be inserted by the editor)

Unsupervised document structure analysis of digital scientific articles

Stefan Klampfl · Michael Granitzer · Kris Jack · Roman Kern

Received: date / Accepted: date

Abstract Text mining and information retrieval in large collections of scientific literature require automated processing systems that analyse the documents' content. However, the layout of scientific articles is highly varying across publishers, and common digital document formats are optimised for presentation, but lack structural information. To overcome these challenges, we have developed a processing pipeline that analyses the structure a PDF document using a number of unsupervised machine learning techniques and heuristics. Apart from the meta-data extraction, which we reused from previous work, our system uses only information available from the current document and does not require any pre-trained model. First, contiguous text blocks are extracted from the raw character stream. Next, we determine geometrical relations between these blocks, which, together with geometrical and font information, are then used categorize the blocks into different classes. Based on this resulting logical structure we finally extract the body text and the table of contents of a scientific article. We separately evaluate the individual stages of our pipeline on a number of different datasets and compare it with other document structure analysis approaches. We show that it outperforms a state-of-the-art system in terms of the quality of the extracted body text and table of contents. Our unsupervised approach could provide a basis for advanced digital library scenarios that involve diverse and dynamic corpora.

Keywords Document structure analysis · Machine learning · Clustering · PDF extraction · Text mining

S. Klampfl · R. Kern
Know-Center GmbH
Inffeldgasse 13/VI
8010 Graz, Austria
Tel.: +43-316-873-30862
Fax: +43-316-873-1030862
E-mail: sklampfl@know-center.at

R. Kern
Knowledge Technologies Institute, Graz University of Technology
Graz, Austria
E-mail: rkern@tugraz.at

M. Granitzer
University of Passau
Passau, Germany
E-mail: michael.granitzer@uni-passau.de

K. Jack
Mendeley Ltd
London, UK
E-mail: kris.jack@mendeley.com

1 Introduction

Managing collections of scientific literature in an automated manner becomes an increasingly important aspect of the daily work of researchers as well as librarians, since the growth of the global volume of scientific literature reaches unprecedented levels. In recent years a number of social research networks, such as Mendeley[1], CiteULike[2], or CiteSeer[3] have emerged that enable users to collect scientific articles and to exchange and discuss them with colleagues.

One important aspect of such systems is the extraction of meta-data, such as title, authors, journal, year, pages, etc., to ease the organisation of scientific literature. But also the automatic extraction of the body text and the table of contents of a given article is of interest for advanced browsing and searching documents in digital libraries. Furthermore, the extraction of named entities and facts contained in the body text and tables is the basis for more complex information retrieval scenarios.

An important prerequisite for all of these tasks is the analysis of the document structure, which is commonly dis-

[1] http://www.mendeley.com
[2] http://www.citeulike.org
[3] http://citeseerx.ist.psu.edu

tinguished into a physical or logical layout [22]. The physical layout of a document refers to its hierarchical organisation into pages, columns, paragraphs, lines, words, and so on, whereas the logical layout consists of a categorization of document parts into meta-data elements, sections, tables, figures, etc. There are two main challenges that need to be tackled: (i) the reliable extraction of text from digital documents, and (ii) the large variety of layouts across different journals and proceedings from different fields. The first challenge is not as trivial as it may sound; the Portable Document Format (PDF), the most common format for scientific literature today, is optimised for presentation, but lacks structural information. It only contains information about individual characters and their position on the page, and this information might additionally be noisy, so intelligent algorithms are required that extract words with correct boundaries in the right order and group these words to lines and contiguous text blocks, which might then be categorized into different types of document parts. Examples for the second challenge are that scientific articles may be arranged in a single column or multiple column format, and that individual elements may appear with different font sizes and weights.

Here we describe a processing pipeline that performs both physical and logical layout analysis from a scientific article in PDF format and uses this information to extract its body text and table of contents. In order to deal with the above challenges we exploit the flexibility provided by unsupervised machine learning algorithms. A demonstration of the system can be accessed online[4], and the source code is available under an open source license[5].

The workflow of our pipeline can be decomposed into individual steps: The first step builds upon the output of the PDFBox[6] library and extracts blocks of contiguous text from the raw PDF file (section 3). This block extraction employs a stack of alternating clustering algorithms to iteratively connect individual characters to words, lines, and blocks. Second, two geometrical relations between text blocks are extracted via graph-based techniques: the reading order and the block neighborhood (section 4). Third, the blocks are categorized into different logical labels based on their bounding boxes and font information (section 5). This categorization stage is based on a combination of clustering and heuristics and also makes use of the geometrical relations above. Finally, we extract the body text, consisting of the section headings and the main text, and use another clustering technique to recreate the table of contents as a tree with the section headings as nodes (section 6).

Both the physical (the extraction of words, lines, and blocks) and logical layout analysis (the categorization of blocks), as well as the extraction of the body text and the table of contents work completely unsupervised and model-free and use only information provided by the current document. In order to obtain a complete processing system of scholarly articles we also included the categorization of meta-data blocks, which we reused from previous work [13]. This approach utilizes a pre-trained supervised classification model, but is completely independent from the rest of the system and only added for the sake of completeness. The unsupervised nature sets our approach apart from a number of related studies that employ supervised classification methods (e.g., [25,4,20]). A performance comparison shows that for a set of articles from the biomedical domain we outperform the approach in [20] in our tasks (section 7).

2 Related Work

Document structure analysis has been a well-studied research problem for a quite some time (see [22] for a review). Early work approached the problem with mostly rule-based systems that operated on scanned document images or the output of OCR. Back then, discovering structure within these documents relied on image processing methods in combination with OCR, such as the XY-cut algorithm [24,23] or the Docstrum algorithm [10] (see [27] for an overview). The former recursively partitions the page image horizontally or vertically at the widest empty rectangles or “valleys”. In contrast to this top-down procedure the Docstrum algorithm finds blocks of contiguous text in a bottom-up manner by finding the nearest neighbors of characters. This approach is very similar to our block extraction algorithm in that it also iteratively connects characters to lines and blocks.

With the advent of PDF as the dominating format for scientific articles, more and more documents were being originally produced in digital form. Many off-the-shelf tools, such as PDFBox, iText[7], JPod[8], or Poppler[9], are able to extract the raw character stream from a PDF document, yet further processing is necessary since the extracted characters might not be in the correct order, or interrupted by decorations (headers, footers, page numbers) or floating objects (images, tables, captions).

The generic extraction of contiguous text blocks from PDF files has usually been tackled by rule-based approaches. For example, a recent algorithm presented in [26] merges words to lines and blocks depending on thresholds estimated from the distributions of word and line distances across each page, while the work of [28] uses a modified Docstrum algorithm to perform block extraction. In contrast to these methods we use clustering to merge characters to words and blocks.

[4] http://knowminer.at:8080/code-demo/index.html
[5] https://www.knowminer.at/svn/opensource/projects/code/trunk
[6] http://pdfbox.apache.org/
[7] http://itextpdf.com/
[8] http://opensource.intarsys.de/home/en/index.php?n=OpenSource.JPod
[9] http://poppler.freedesktop.org

A number of recent articles also tackled the analysis of the logical document structure of PDF files. The aforementioned paper [26] describes an open source system for analysing PDF publications in the biomedical domain. This system uses heuristics to extract text blocks from the PDF and a rule-based method to classify these blocks into "rhetorical" categories. This categorization stage achieves a very good overall performance, but requires the user to specify a separate rules file for every different journal layout, which contains for every block class the necessary conditions on layout and formatting. The authors also evaluate the main text flow, but do not detail any efforts in determining the reading order.

The authors of [9] present a comprehensive system for the structure extraction of PDF books, which is used within a commercial e-book software. They perform a categorization of text blocks through a combination of heuristics, clustering, and supervised learning. Their approach is rather similar to ours, in particular, we build upon the same decoration detection method [16]. They calculate the reading order of blocks by computing the optimal matching in a bipartite graph, using not only positional information, but also the rendering order and the text content of blocks. Furthermore, they extract the document hierarchy from the book's table of contents section in a rule-based manner.

Another document processing system, DOMINUS [7, 8], also intensively applies machine learning in multiple stages, for example, to the extraction of text blocks and to the categorization of meta-data. Among the techniques employed are Markov Logic Networks and Inductive Logic Programming, which aim at the automatic generation of rules or theories from a number of examples. These rules control whether document objects are merged or split, and how they are categorised. The main advantage of these methods in the context of digital libraries is their incremental nature that allows them to adapt to novel observations without the need to retrain a model in a batch manner.

Other work targeted the extraction of certain aspects of PDF documents, such as meta-data [12] or tables [19]. A popular supervised learning method for structure analysis are Conditional Random Fields (CRFs) [25]. One example is the ParsCit system [4], which uses a combination of heuristics and CRFs for reference parsing. A related system is SectLabel [20], which builds upon the feature sets defined for ParsCit to detect the logical structure of whole scientific documents, and which categorizes the individual lines of a raw input text file. We use this system for comparison in our evaluation (section 7).

3 Extracting contiguous text blocks

The first part of our processing pipeline builds upon the output of the PDFBox library and consists of the unsupervised extraction of words, lines, and contiguous text blocks from the raw characters. The output of PDFBox is a list of characters, their bounding boxes (x and y position on the page, as well as their width and height), and information about their font. For the conversion to plain text PDFBox also uses mechanisms for detecting word, line, and paragraph boundaries, which are based on simple heuristics depending on the relative position of neighboring characters. However, we decided to build our own generic text block extractor and did not reuse existing approaches provided by PDF parsing libraries, mainly because we want to leave open the possibility to apply our pipeline also to other input formats, for example the output of OCR software. Another reason for not using PDFBox for the extraction of text blocks is that it does not provide any geometric information about these compound objects, which we need at later stages in our pipeline. It might also be desirable to extend our block extractor by incorporating font information or special rules such as the splitting of words at superscripts or subscripts.

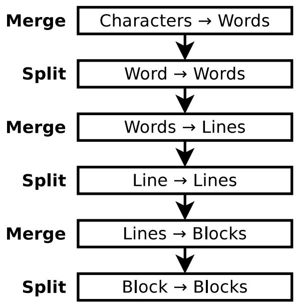


Fig. 1 The stack of alternating clustering algorithms used for extracting text blocks from a PDF in a bottom-up manner. The *Merge* steps use hierarchical agglomerative clustering (HAC); in the *Split* steps standard k-means clustering is employed.

The main challenge in the extraction of text blocks is that the information provided by PDFBox might be unreliable: for example, height and width information might be slightly wrong, or information about the font of some characters might be missing. We therefore require algorithms which are flexible enough to deal at the same time with both this noisy data and the variety of layouts of scientific publications.

Here, we chose methods from unsupervised machine learning, in particular clustering, to solve this task. More precisely, we used a sequence of *Merge* and *Split* steps to iteratively combine individual characters to words, lines, and blocks of text in a bottom-up manner (Figure 1). Each *Merge* step is implemented by hierarchical agglomerative clustering (HAC) with Euclidean distance measure and Single Linkage. In the first *Merge* step individual characters are merged to words: pairs of characters with increasing distance to each other are combined into clusters, until a maximum distance threshold is reached. Since the resulting clusters of characters might now encompass multiple words, a *Split* step is incorporated in the form of standard k-means clustering on the horizontal distances between characters ($k = 2$). Ideally, this partitions the spaces between characters into spaces between words and spaces within words, yielding the final set of words. This *Split* step can also be understood as an outlier detection which removes too large inter-character distances from the words obtained in the *Merge* step.

Another pair of *Merge* and *Split* steps was used to combine words to lines and lines to blocks. First words are merged to lines by combining pairs of words with increasing Euclidean distance to each other. This typically yields lines spanning multiple columns, which is resolved in the *Split* step that separates word spaces within columns from inter-column spaces. Finally, lines are merged to blocks, again by first combining them until a maximum distance threshold is reached, and then by splitting the resulting clusters at large vertical distances. Examples of extracted text blocks are shown in Figure 2.

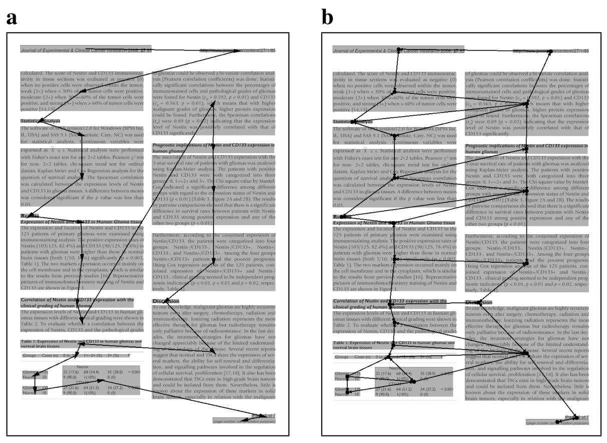


Fig. 2 Examples of extracted block relations. Both panels show the same document page with the extracted text blocks. (a) The reading order is a permutation of the blocks on a page in which they are supposed to be read by humans. Here, the reading order is determined for all blocks on the page, in a later stage it is post-processed to contain only the body text. (b) The block neighborhood indicates the nearest neighbors in each of the directions top, bottom, left, and right.

4 Extracting relations between text blocks

We consider these blocks of contiguous text extracted in the previous section as the basic elements of a scientific document. In the next step of our processing pipeline we extract two geometrical relations between the text blocks on a page that serve as additional information in the categorization stage. The first relation is the *reading order*, the order in which blocks on a page are supposed to be read by humans, the second is a simple geometrical neighborhood relation.

4.1 Reading order

In Western culture and for most scientific documents the reading order is column-wise, i.e., text is read from top to bottom in columns, which are then read from left to right. Detecting the reading order of text blocks is a non-trivial problem, since the text flow might be interrupted by figures and tables and their captions spanning multiple columns. Additionally, text blocks inside figures and tables, as well as footnotes and decorations such as page numbers or headings, are often not aligned to these columns.

The reading order on a given page is defined as a specific permutation of all text blocks on that page. We first determine the reading order of all blocks on the page, regardless of whether their content belongs to the main text of the document. This information is later used for the categorization of blocks. Afterwards the reading order is postprocessed to retain only those blocks which belong to the body text of the document.

We follow the approach of Aiello et al. [1], who defined a set of binary relations for intervals in X and Y direction that allow a certain amount of tolerance for the coordinate values. In total there are 13 relations in both X and Y direction, and for each pair of two-dimensional bounding boxes exactly one X relation and exactly one Y relation is true. This tolerance is implemented by a parameter T; if two coordinates are closer than T they are considered equal. This flexibility is necessary because due to the inherent noise in the PDF extraction text blocks in the same column might not be exactly aligned (here we choose $T = 5$). Aiello et al. then defined the *BeforeInReading* relation as a Boolean combination of binary relations for intervals in X and Y direction, which states for any pair of bounding boxes whether the first one occurs at some point (not necessarily immediately) before the other in a column-wise reading order (see Figure 5 in [1] for the exact definition).

In addition to [1], we also define the *BeforeInRendering* relation that tells whether a block is rendered at some time before another block in the PDF. We incorporate both relations into a single partial ordering of blocks by specifying a directed graph with an edge between every pair of blocks for which at least one of the two relations hold. We then perform topological sort on this graph by sorting the nodes by the number of outgoing edges in descending order; the first node in this sorted list is the first node in the reading order on that page. We remove that node and all edges connecting this node, resort the nodes by the number of remaining outgoing edges, and select the next node for the reading order. This is repeated until all nodes of the graph have been removed, yielding a permutation of the blocks on the page as the reading order.

Figure 2a shows an example document page with the extracted reading order of all blocks on that page. It also shows an example where the rendering order is relevant. Based on geometrical information only the table on the bottom left of the page would be placed in the reading order between the text blocks of the left and the right column. But since the table is rendered separately from the main text on that page, utilizing this information results in the more convenient placement at the end of this page.

The presented method for detecting the reading order differs from the approach in [1] in some aspects. First, we did not use their special topological sort variant for transitive directed graphs because we found that even the *BeforeIn-Reading* relation alone does not necessarily yield transitive graphs.[10]. We therefore use the simplified topological sort variant described above. Second, we also report only one reading order even when there are multiple equally good ones possible. More sophisticated methods for reading order detection have been defined [9,21], but we found that our simplifications yield satisfying results for scientific documents and once the reading order is restricted to blocks containing the main document text in a later stage.

4.2 Block neighborhood

The second geometrical relation between blocks that we extract as a preprocessing step is the block neighborhood. Examples where this relation is useful include headings, which should be above another main text block, or tables and figures, which should be neighbors to a caption block. We employ a simple straightforward algorithm that searches for the nearest neighbor of each block on the page in each of the four main directions, viz., top, bottom, left, and right. This yields a directed neighborhood graph of blocks on the page, since this relation is not necessarily symmetric (e.g., a heading block in the left column that is shorter than the main text width might have a text block in the right column as its nearest neighbor to its right, however, the leftmost neighbor to that block is another text block in the left column). An example page with extracted block neighborhood relations is shown in Figure 2b.

For our setup, we found this neighborhood relation more usable than using a Voronoi diagram computed for the centres of the blocks (as in [1]) because the latter often results in blocks being connected that are quite distant to each other. Especially small blocks like page numbers or short headings tend to share edges with more distant blocks because of the horizontal offset of their centers, but that might depend on the granularity of the text block extraction stage.

[10] Consider a page with four text blocks arranged in two columns (two blocks in each column) and in the middle of the page there is another block spanning both columns. Then the top right block is before the middle block in the reading order, the middle block before the bottom left block, but the bottom left block before the top right block.

5 Categorization of text blocks

The categorization of text blocks is implemented as a sequential pipeline of *detectors* each of which labels a specific type of block: decorations (such as page numbers, headers, and footers), figure and table captions, main text, section headings, sparse blocks and tables. Each of these detectors is completely model-free and unsupervised. We derive the categories only from information provided by the current document: they only use the labels given by previous detectors, the geometric information of the text blocks, their content including font information, as well as the block relations extracted before and described in the previous section. For the sake of completeness we also added a meta-data detector, which is based on previous work [13] and uses a pre-trained supervised classification model, but is completely independent from the rest of the system.

5.1 Decorations

Many digital documents have archival information such as author names, publication titles, page numbers, and release dates printed repeatedly at the border of each page. Most prominently this content is placed inside headers or footers, but sometimes also at the left or right edge of the page. We refer to text blocks containing this type of information as decoration blocks.

We adopt the work in [16], which is based on associating top and bottom lines across neighboring pages based on both their content and their position on the page. This is considered one of the standard header/footer detection methods [9]; other work relies upon the presence of specific visual cues such as horizontal lines [15], which is not the case for many of the numerous layouts used for typesetting scientific articles.

We use a slightly modified variant of the approach in [16] that is applicable to text blocks instead of lines. The block extraction stage does not necessarily extract single header and footer lines, rather there are usually individual blocks for the individual elements such as the page number and the journal. We proceed as follows. First, for each page, we sort all blocks on the page in four different orders: from top to bottom (based on the minimum y coordinate), from bottom to top (maximum y coordinate), from left to right (minimum x coordinate), and from right to left (maximum x coordinate). Separately for each ordering we compute the following similarity score for each of the first N blocks:

$$\mathrm{Score}(B) = \max_{i=1,\ldots,N} \left\{ \frac{1}{2} \left(\mathrm{Similarity}(B, B_{n,i}) + \mathrm{Similarity}(B, B_{p,i}) \right) \right\}, \tag{1}$$

where $B_{n,i}$ and $B_{p,i}$ are the i-th blocks in the same ordering on the next and previous page, respectively. These relations are circular, i.e., the first page is next to the last page. If the document has more than three pages, we compare blocks on the next or previous page with an even or odd number, depending on whether the current page number is even or odd, to account for cases with a two-sided layout.

The similarity of two blocks in (1) is measured both geometrically and based on their content,

$$\text{Similarity}(B_1, B_2) = \text{ContentSimilarity}(B_1, B_2) * \text{GeomSimilarity}(B_1, B_2). \quad (2)$$

This similarity score is a value in the range $[0, 1]$ and given by the product between the content and the geometric similarity. The former is calculated from the normalized edit distance between the two content strings, where digits are replaced with "@" chars. A content similarity of 1 is reached when both strings are exactly equal. The geometric similarity is the area of the intersection between the two bounding box rectangles divided by the larger of the two bounding boxes.

A block is labelled as *Decoration* if its score exceeds some predefined threshold Θ_{score}. Here, we choose $N = 5$ and $\Theta_{score} = 0.25$. The relatively low threshold allows for some noise in the block extraction stage, for example, on different pages of the same document headers might be extracted as a single or multiple blocks. Examples of blocks labelled as decorations are shown as red blocks in Figure 4, at the top and bottom of the example pages.

5.2 Captions

Captions are text blocks usually located directly above or below a figure or table explaining its contents. We first detect candidates for caption blocks by simply checking whether its first word equals one of certain predefined keywords (viz., "Table", "Tab", "Tab.", "Figure", "Fig", "Fig.") and the second word contains a number (optionally followed by a punctuation, such as ":" or "."). This simple heuristic has been found sufficient for previous work [19,9].

The caption candidates should now contain the true captions, but also blocks from the main text that accidentally start with one of the keywords (e.g., "Table 1 shows that..."). In order to remove these spurious captions, we analysed the formatting of the candidates. For each block we determined the font size and the optional punctuation character, and removed each block that does not conform to the majority values of these features across all candidate blocks.

Examples of categorized captions can be seen as cyan blocks in Figure 4.

5.3 Main text

The main text of a scientific document is typically organised into one or multiple columns, structured into sections, and might be interleaved with tables or figures. Here, we aim to detect to all blocks containing the main text of the document apart from the section headings, which are detected in the next section.

The main characteristic that we exploit here is that main text blocks share the same alignment and formatting (cf., blue blocks in Figure 4). More precisely, we identified the following properties of text blocks containing the main text of most scientific articles:

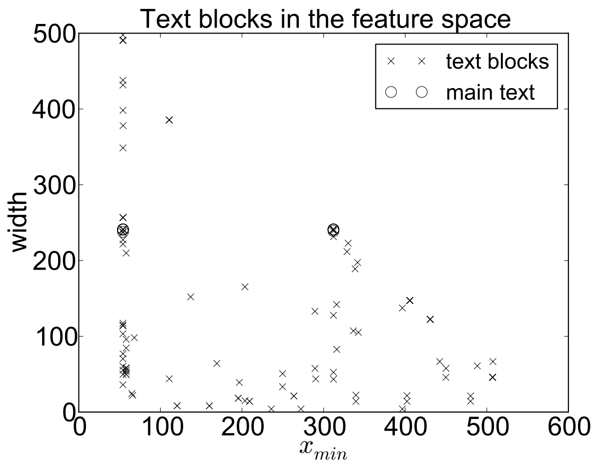
i) they are left-aligned to a limited number of x coordinates (typically the number of columns)
ii) they have a similar width (if the text is justified, the width is virtually identical)
iii) the font of the majority of characters inside the block is the same for all main text blocks, and
iv) the majority of lines in the document belong to the main text.

In order to capture the similarities expressed by properties i) and ii), we applied hierarchical agglomerative clustering (HAC) on all blocks of a document in the two-dimensional feature space defined by the left x coordinate and the width of the blocks. As inter-cluster distance we used "single link"; as the distance between two blocks we used standard Euclidean distance, however, for two blocks with a different majority font we set the distance to positive infinity. This accounts for property iii) and basically ensures that such pairs of blocks end up in different clusters, or, equivalently, all blocks inside one cluster share the same majority font.

Figure 3a shows all text blocks of a typical scientific article with a two-column layout in the two-dimensional feature space used for clustering. The desired main text is shown as circles, forming two clusters, one for each column. Note that here multiple blocks overlap each other. This emphasizes the need for a distance based clustering algorithm such as HAC, as opposed to, say, k-means, which would not be able to separate main text blocks from other blocks close by.

HAC merges blocks bottom-up with decreasing similarity and stops once a distance threshold is reached. We chose 10 as the distance threshold; it should be large enough to allow for some variability for the alignment of blocks inside a column, but small enough not to merge blocks across columns or very short blocks. In the next step we sort the resulting clusters by their total size in *lines* in decreasing order, such that according to property iv), the largest clusters should contain the main text blocks (see Figure 3b). We iterate over this sorted list of clusters and label the contained blocks as *Main text*, until we encounter either a large change in the average width of blocks (larger than a threshold $\Theta_{width} = 5$), or a simultaneous change in font size and

a



b

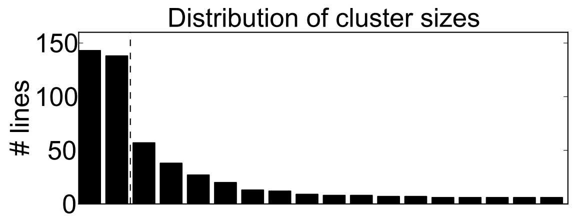


Fig. 3 (a) All text blocks of a typical scientific article represented in the two-dimensional feature space spanned by their minimum x-coordinate and their width. Each cross represents one text block; blocks categorised as main text are surrounded by a circle. Note that multiple text blocks are on top of each other. (b) Distribution of cluster sizes resulting from HAC, measured as the number of lines in all blocks of that cluster in descending order. The vertical dashed line indicates a cut defined by a large change in cluster size. All blocks in the clusters to the left are labelled as main text.

cluster size (the portion of all document lines changes by more than $\Theta_{size} = 0.1$). The reason for the second criterion is that main text may consist of more than one font size (e.g., methods sections in biomedical papers), but we include blocks of different font size only if there is a substantial amount of text.

The flexibility of the clustering algorithm deals with small disalignments, e.g., slightly indented blocks such as enumerations or lists, but some layouts might require a tuning of the threshold parameters. Main text blocks remain undetected mainly if their width substantially deviates from the normal column width, e.g., when text floats around a figure spanning 1.5 columns, or when the main text spans the whole page width on the first page of the paper, while being set in two columns for the remaining part. We also do not handle equations or other elements that are embedded into the main text flow.

5.4 Headings

The main text of a scientific article is typically hierarchically structured into sections and subsections. Here we aim at extracting all text blocks containing these section headings irrespective of their level in this hierarchical structure. Later, in section 6.2, we try to reconstruct this structure in the form of a table of contents.

The detection of these section headings is based on the previous labelling of main text blocks and uses additional information provided by the block relations, reading order and block neighborhood. A necessary condition for a text block to be considered as a heading is that it occurs either immediately before a main text block in the reading order or is the top neighbor of a main text block. The reason for incorporating both relations is that in some cases a heading does not occur immediately before a main text block, even if it is the direct top neighbor. Furthermore, a candidate heading block has to be either left- or centre-aligned to the following main text block. This is checked by testing if the left or center x coordinates of the bounding boxes agree within some tolerance T (here, $T = 5$). Additionally, each of the following conditions must be met:

i) the text starts with either a number or an uppercase character,
ii) apart from an optional numbering it consists of at least one non-whitespace letter,
iii) it has a maximum number of lines (here: 3),
iv) the majority font size is at least as large as for the neighboring main text block,
v) the distance to the neighboring text block is lower than a threshold (here: 4 times the size of a line of the current heading block candidate).

If these conditions are satisfied for unlabelled blocks, they are labelled as *Heading*. A main text block is allowed to be relabelled, if in addition the majority font is bold or italic.

Headings can not only occur before a main text block, but also immediately before another heading (in the case of headings with different levels). To account for these cases we repeat the detection algorithm and check blocks immediately before or directly above another heading block. By varying the number of passes of the algorithm over all blocks of the document we can control the maximum number of directly adjacent headings. Here, this value is set to 2.

Examples of detected headings are shown as green blocks in Figure 4.

5.5 Sparse blocks and tables

The remaining blocks contain various document elements such as references, footnotes, formulas, and texts in figures and tables. Most of these blocks share at least one of two

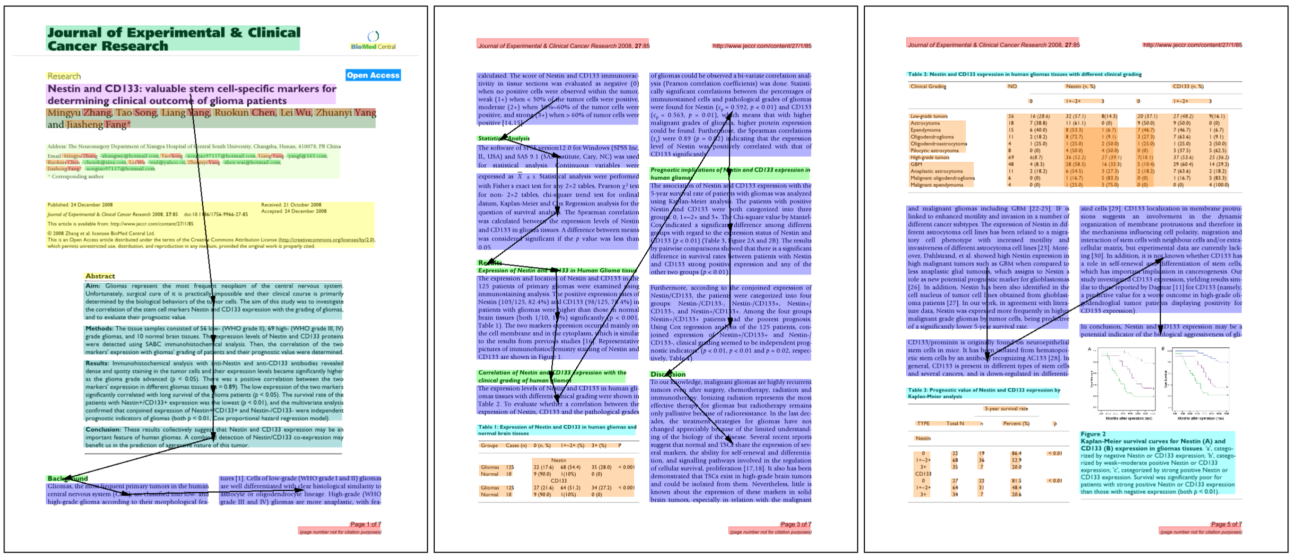


Fig. 4 Example pages with text blocks categorized into different classes (denoted by different colors): decorations (red), captions (cyan), main text (blue), headings (green), sparse lines (yellow), and tables (orange). The first page contains meta-data blocks labelled with different colors. Furthermore, the postprocessed reading order is shown, containing only those blocks that are part of the body text (headings and main text).

properties: (1) their width is substantially smaller than the column width, or (2) there exists a gap between two consecutive words in the block that is larger than the average width between two words in the main text. These properties have been identified in [19] as conditions for so-called *sparse lines*, a subset of the document relevant for table detection. There the authors designed an algorithm for constructing lines from the characters in the PDF and defined a line to be sparse if either of the above conditions was met. We applied the same conditions to our text blocks, except that we calculated the average column width and average word space only from blocks previously categorized as main text. We labelled a block as sparse if its width was smaller than 2/3 of the average width of a main text block, or it contained a gap between two consecutive words larger than two times they average gap between words in main text blocks. To account for corner cases we also relabelled main text blocks as sparse blocks if their *average* word gap was larger than two times the document average. These thresholds were taken from [17].

Furthermore, we implemented a table boundary detection algorithm similar to that presented in [19]. Starting from a table caption (recognized by checking if it starts with a keyword such as “Table”) neighboring sparse blocks (defined by the block neighborhood) are labelled as table blocks if their vertical distance does not exceed a threshold of 2.5 times the average line height of the document main text. Restrictions on the horizontal distance are only used if we identify the current table as a single-column table, which is the case if the caption block is left- or center-aligned to a column and its width is smaller than the column width. For a single-column table the labelling stops once a candidate sparse block does not lie completely inside the current horizontal column borders.

The example pages of Figure 4 show 3 tables where blocks have been successfully labelled as table blocks (orange). The first page contains a block that has been categorised as sparse (yellow) because it contains large gaps between words.

Table boundary detection, identifying those text blocks which belong to a table, is an important preprocessing step for parsing the structure of the table and extracting its content. This is a subject of future work and beyond the scope of this paper.

5.6 Meta-Data

In a scientific document specific text blocks contain meta-data information about the published article, e.g., the title, the journal in which it was published, or the abstract. To detect these meta-data blocks, we reused previously published work [13] which employs a supervised classification approach that additionally takes the sequence of labels into account. The meta-data blocks are categorised into the following types: *Title*, *Journal*, *Author*, *Affiliation*, *Email*, and *Abstract*. The details of this approach are beyond the scope of this paper; the interested reader is referred to [13]. The top left panel in Figure 4 shows an example page with text blocks categorised into different types of meta-data. For author type blocks the contained tokens are further classified into given names, surnames, emails, and affiliations.

List of headings in the reading order:

- Background
- Materials and methods
- Patients and Tissue Samples
- Methods
- Immunohistochemical Staining
- Statistical Analysis
- Results
- Expression of Nestin and CD133
- Correlation of Nestin and CD133 expression
- Prognostic implications of Nestin and CD133
- Discussion
- Abbreviations
- Competing interests
- Authors' contributions

Clusters with assigned heading level:

Heading level 1:

- Background
- Materials and methods
- Methods
- Results
- Discussion
- Abbreviations
- Competing interests
- Authors' contributions

Heading level 2:

- Patients and Tissue Samples
- Immunohistochemical Staining
- Statistical Analysis
- Expression of Nestin and CD133
- Correlation of Nestin and CD133 expression
- Prognostic implications of Nestin and CD133

Resulting table of contents:

- Background
- Materials and methods
  - Patients and Tissue Samples
- Methods
  - Immunohistochemical Staining
  - Statistical Analysis
- Results
  - Expression of Nestin and CD133
  - Correlation of Nestin and CD133 expression
  - Prognostic implications of Nestin and CD133
- Discussion
- Abbreviations
- Competing interests
- Authors' contributions

Fig. 5 This example shows the workflow of our algorithm for extracting the table of contents of a specific scientific article from the biomedical domain. First, all headings are collected (left) and clustered based on their formatting. The resulting clusters are then sorted yielding a heading level for each cluster (middle). Finally, the table of contents are generated by creating a tree (right). The headings are processed in the reading order and each heading is assigned as a child to the last heading with a higher level (right).

6 Body text and table of contents extraction

Once the text blocks have been categorized we can now extract the body text and the table of contents of the given document. The body text is given by the sequence of section headings and main text blocks in the extracted reading order. The table of contents is determined by creating a tree structure of headings using hierarchical clustering.

6.1 Body text extraction

The first step in extracting the body text from the document is to post-process the reading order, which has been originally determined from all blocks, to contain only section headings and main text blocks. This is achieved by a straightforward sequential filtering (see Fig. 4). Additionally we remove sections titled "Abstract", "Acknowledgments", "References", "Bibliography", or "Supporting Information" from the body text: once we encounter a heading with this content we remove this block and all immediately following main text blocks until the next heading block. The body text is then composed by a simple concatenation of the contents of the remaining blocks in the sequence of the reading order.

In many documents words are hyphenated if they do not fit on the current line, especially if the text layout is justified. However, the hyphenation chars "-" should not be part of the extracted body text. We resolve hyphenations by removing hyphens "-" and concatenating the split word parts if they are the result of a proper English hyphenation. For each line of a main text block that ends with a hyphen we apply the hyphenation on the concatenated word using a list of hyphenation patterns taken from the TEX distribution, and if the line split occurs at one of the proposed split points we resolve the hyphenation. As we repeat this check across main text blocks, hyphenations are also resolved across columns and pages.

6.2 Table of contents extraction

The task of the table of contents (TOC) extraction is to recreate the structure of the scientific article and to identify the hierarchy of sections. The output of this process is a tree of headings for the individual sections. We use the blocks labelled as headings as the starting point for the TOC extraction. Our approach is divided intro three stages:

i) grouping of headings into different levels based on their formatting,
ii) determining the heading levels based on ordering the resulting groups, and
iii) using the sequence information within the article and the ordering of groups to create a hierarchy.

The approach completely relies on unsupervised methods, therefore there is no need for training examples, and it also should work independently of domain.

In the first stage of the algorithm we group headings of similar formatting. We use the HAC clustering algorithm, where the distance function is a weighted sum of the differences of the mean character height and of the mean number of characters of two clusters. More precisely,

$$d(c_1,c_2) = \frac{|\overline{h_{c_1}} - \overline{h_{c_2}}|}{\min(\overline{h_{c_1}},\overline{h_{c_2}})} + 0.1\frac{|\overline{l_{c_1}} - \overline{l_{c_2}}|}{\max(\overline{l_{c_1}},\overline{l_{c_2}})}, \tag{3}$$

where $\overline{h_{c_i}}$ denotes the average height of a character and $\overline{l_{c_i}}$ the average character count of a heading in cluster c_i. The distance between two clusters is set to infinity if one of the following criteria is met:

i) two headings are directly adjacent,
ii) either one of the clusters is made up exclusively upper-case characters,
iii) the difference in mean character heights differs by more than $\max(stdev(h_{c_1}), stdev(h_{c_2})) + 0.01$, or
iv) the level of the numbering that precedes the heading text differs.

The input for the second stage is the list of clusters that contain at least a single heading. These clusters are ordered according to their assumed heading level using a pairwise comparison function which outputs $\{-1, 0, +1\}$ to indicate the ordering. This comparing function operates according to the following precedence rules:

i) number of prefix segments,
ii) difference in mean character height, and
iii) preference of all upper-case clusters.

The output then is a sequence of headings where the ranking defines the heading level.

The final stage takes the ranked list and produces the final TOC tree by using the information provided by the extracted reading order. Starting from an empty tree with a single root node all headings are iterated in the sequence of how they appear within the article. The first heading is added as a child to the root node. For every following heading we search backwards until we find a heading with a higher heading level, and add it as a new child to this parent heading. If no such parent heading is found, we add the heading as a new child to the root node. By enforcing a valid tree hierarchy this approach corrects some errors made during the first two stages.

7 Evaluation

In this section we present the evaluation of our processing pipeline. First, we evaluate the quality of both the block extraction and the block categorization on the GROTOAP dataset, which consists of labelled segmentations of scientific documents into zones. Next, the performance of the main text and heading extraction is assessed on the PubMed dataset, which provides a structured XML file along with each PDF document. The main text evaluation is based on a modified edit distance and indirectly measures the quality of the detected reading order. Finally, we determine the correctness of the extracted tables of contents on the same dataset using an edit distance measure on trees.

7.1 Block extraction

A ground truth dataset for evaluating the segmentation of document pages into contiguous text blocks has recently been made available: the GROTOAP dataset [29] consists of 113 documents from various open access journals in digital form as well as their geometric hierarchical structure in XML format. For each word, line, and zone (as the analogue to a block is called there) the corresponding bounding box is given in terms of its x and y coordinates on the page. In contrast to previously available datasets which are usually based on scanned document images, the GROTOAP dataset is to the best of our knowledge the first that provides this information for born-digital documents. In a related paper the same authors use this dataset as part of a similar metadata extraction process [28], however consisting of a different block extraction method, a modified Docstrum algorithm [10]. In the following, we present the evaluation of our block

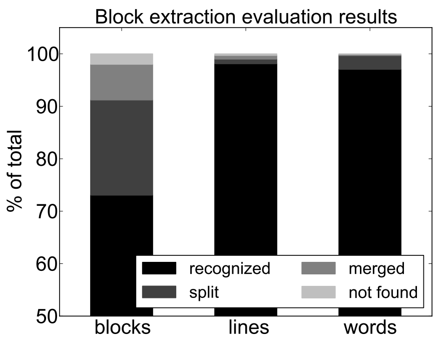


Fig. 6 Evaluation of our approach to block extraction on the GROTOAP dataset. The bar plot shows the percentage of ground truth objects that were recognized, split, merged, or not found, respectively.

Table 1 Detailed evaluation of the block extraction stage across all 113 documents of the GROTOAP dataset and comparison to the approach provided by the PDFBox library.

Our approach:

	total	**recognized**	**split**
blocks	8308	6070 (73.06%)	1508 (18.15%)
lines	65361	64124 (98.11%)	582 (0.89%)
words	560012	543491 (97.05%)	14650 (2.62%)
		merged	**not found**
blocks		565 (6.80%)	165 (1.99%)
lines		457 (0.70%)	198 (0.30%)
words		1205 (0.22%)	666 (0.12%)

PDFBox:

	total	**recognized**	**split**
blocks	8308	5040 (60.66%)	1799 (21.65%)
lines	65361	63353 (96.93%)	665 (1.02%)
words	560012	557385 (99.53%)	1506 (0.27%)
		merged	**not found**
blocks		1465 (17.63%)	4 (0.05%)
lines		1333 (2.04%)	10 (0.02%)
words		1050 (0.19%)	71 (0.01%)

extraction algorithm pipeline on this dataset and also compare our results to that paper.

We followed the approach in [28] and determined for each object in the ground truth (word, line, or zone) whether it was correctly recognized by our block extraction algorithm, or whether it was split, merged, or not found at all. This decision was made by comparing the sets of non-whitespace characters that the corresponding objects contained. We ignored whitespace characters in this comparison to account for the fact that in some PDF files spaces between words are not expressed as separate characters. In order to determine the corresponding extracted object for a given ground truth object in the first place, we applied a naive search procedure to determine that object of the same type (word, line, or zone/block) that has the maximum geometric overlap to the ground truth object. This is reasonable in terms of computational efficiency since we can restrict the search to objects on the same page.

Given character sets S_T and S_E contained by the ground truth and extracted objects, respectively, we applied the following rules to determine the segmentation outcome:

condition	outcome
$S_T = S_E$	recognized
$S_T \subset S_E$	merged
$S_T \cap S_E = \emptyset$	not found
else	split

An object is considered to be recognized if both character sets are exactly the same. If the set of characters in the ground truth object is a proper subset of the characters in the corresponding extracted object it is said to be merged. On the other hand, a split occurs if characters of one object in the ground truth belong to more than one extracted object. It is worth noting that with this evaluation procedure the merging of n original zones to one extracted block counts as n merges, however, the splitting of a single ground truth zone into n blocks accounts for only one split. Finally, it may also happen that no corresponding (i.e., overlapping) extracted object can be found, or the sets of characters are completely disjoint. In most cases this can be attributed to errors in the underlying text extraction software, which sometimes yields wrong characters or bounding boxes (e.g., rotations applied to labels in figures). Moreover, visual inspection showed that many incorrectly recognized objects occur inside tables and figures, where neighboring table cells might be merged to single lines, axis labels of figures to single words, or blocks might span individual table cells or complete columns or tables. We therefore excluded all ground truth zones with one of the labels *table*, *figure*, or *unknown* from the following analysis.

The results of the block extraction evaluation are shown in Figure 6 and Table 1. The fraction of correctly recognized lines and words are both above 97%. Also the majority of blocks is recognized, however, the numbers of splits and merges are relatively high. For example, headers or footers were sometimes merged with adjacent page numbers or related decorations. The high number of split blocks mostly results from the fact that our block extraction algorithm tends to split long blocks of main text into multiple parts, especially when there is a increased line distance due to a superscript or subscript. Furthermore, in the ground truth dataset headings were often combined with the succeeding main text paragraph. In our pipeline these two parts were deliberately split in order to endow further post processing such as table of contents extraction.

A closer inspection of the GROTOAP dataset further revealed that individual words in the ground truth sometimes contained adjacent punctuations, like ".", ",", "(", or ")", and sometimes did not, while our block extraction usually yielded only the actual word. To account for these spurious splits and merges we ran another evaluation where we compared only character sets consisting of letters and digits, resulting in 98.14% correctly recognized words, 98.60% lines, and 74.35% blocks. Similar effects occurred for superscripts, footnote marks, and apostrophes. Another crucial issue concerned the correct handling of ligatures "fi" or "fl", which appeared as single characters in the ground truth, but were correctly normalized by our processing pipeline. Diacritic characters, such as umlauts or letters with accents, were another source of spurious splits since they were sometimes represented as a separate pair of characters in the ground truth.

We also evaluated the heuristics that PDFBox provides to detect words, lines, and paragraphs during the conversion of PDF into plain text. Based on the statistics of horizontal gaps between characters it inserts spaces between words, if necessary; it checks for line breaks based on the vertical overlap of neighboring characters, and inserts paragraph breaks based on the distance of lines and their indentations. At this stage PDFBox provides no geometrical information about these compound objects, so instead for searching the block, line, or word with the maximum geometric overlap, we performed the evaluation based on text content alone. For each zone, line, or word in the ground truth we determined the paragraph, line, or word that had the greatest overlap in terms of the longest common subsequence of their textual contents. If there were more than one candidates with the same overlap we selected the one with the smallest edit distance. This resulted in the correct recognition of 60.66% of the zones, 96,93% of the lines, and 99,53% of the words. The recognition rate for zones and lines is lower than for our block extraction algorithm (see Table 1). One reason for the low performance on zones is that PDFBox extracts individual paragraphs, which typically have a finer granularity than the ground truth zones.

Table 2 Contingency table of blocks labelled as meta-data evaluated on the GROTOAP dataset. Columns correspond to our labels, rows correspond to GROTOAP labels. Boldface entries denote equivalent labellings.

	Title	*Journal*	*Author*	*Affiliation*	*Email*	*Abstract*
abstract	1	0	0	0	0	**195**
body	0	0	0	0	0	3
correspondence	0	0	0	0	**31**	0
unknown	0	0	0	1	0	0
copyright	0	0	0	0	6	0
author	0	0	**102**	0	1	0
title	**109**	0	1	0	0	0
dates	0	0	0	0	1	0
bib_info	0	**1**	0	0	0	0
affiliation	1	0	0	**96**	1	0
Total	111	1	103	97	40	198

In [28] the authors reported that 90.74% of the zones, 98.56% of the lines, and 99.49% of the words were recognized correctly. However, the above considerations suggest that the achieved performance very much depends on the exact definition of zones or blocks, lines, and words (e.g., does a period belong to the preceding word? Is a table a single block? Does a heading belong in the same block as its successive paragraph?). Also the comparison of objects based on their characters requires a correct encoding of these characters. Another possibility, which we did not pursue further, would be to compare segmentation objects based on their geometric information (i.e., bounding boxes) only.

7.2 Block categorization

Since the GROTOAP dataset also contains a labelling of each block, we evaluate the block categorization on the same dataset as the block extraction in the previous section. This dataset provides a rather fine-grained labelling of the various text blocks (see row headings in Tables 2 and 3). Since their block extraction method is different from ours it sometimes yields a different granularity of text blocks: blocks might be merged or split compared to the ground truth segmentation. For the alignment of our extracted labels with the ground truth labels we choose the simple strategy that for each extracted block we search for the corresponding ground truth zone that has the maximum overlapping area and compare the two labels. Note, that this might result in smaller ground truth zones not being used for comparison at all, or larger zones to be compared multiple times.

The resulting contingency tables are shown in Table 2 for the meta-data blocks and in Table 3 for the remaining blocks. It can be seen that the meta-data extraction, which is based on a supervised classification problem [13], achieves a good performance since most blocks have a corresponding ground truth zone with a similar label. Decoration blocks are mostly paired with zones labelled as *page_number* or *bib_info* (usually the journal name or other publishing information), which typically occur in headers or footers of documents. Decorations are mislabelled if a block with similar content accidentally repeats at almost the same position on neighboring pages (e.g., equation numbers, table elements). Caption blocks are mostly assigned to figure and table captions; blocks of the body text are sometimes erroneously labelled if their text starts with one of the caption keywords. Main text blocks largely correspond to zones labelled *body*; depending on the font size, the reference section might be part of the main text or not. Most headings also overlap with a *body* zone; the reason for that is that in the ground truth dataset headings are typically merged to the *body* zone of the following paragraph. A large part, but not all of the sparse blocks inside tables have been correctly identified as table blocks, indicating room for improvement by further work on table recognition. As expected, the remaining sparse blocks are mostly composed of small text blocks inside figures, equations and their labels, and parts of the main text which are aligned differently from the standard columns, e.g., lists or insets.

7.3 Body text and headings

For the evaluation of the quality of the body text and headings extraction we use a dataset of 1000 randomly selected documents from PubMed[11], a free database created by the US National Library of Medicine holding full-text articles from the biomedical domain together with a standard XML markup that rigorously annotates the complete content of the published document, in particular the metadata, the section headings, and the body text. The documents contained in this database are very diverse. Most of the documents are research articles, but there is also a wide range of different article types, including book reviews and meeting reports. Table 4 shows the distribution of article types for the selected subset of 1000 documents. Furthermore, the documents vary strongly in their layout, ranging from more book-style single column layouts to the standard two-column layouts of scientific articles and layouts with three or more columns of review or news articles. In total, the 1000 selected documents are published in 220 unique journal titles; the most frequently occurring journal titles are shown in Table 4.

First, we evaluate the quality of the extracted main text (the content of heading and main text blocks in the reading order, see section 6) by comparing it to the concatenated string of characters contained in the body part of the ground truth XML. We remove all whitespace characters in both strings and determine their similarity by a variant of the Levenshtein distance that counts the number of insertions and

[11] http://www.ncbi.nlm.nih.gov/pubmed/

Table 3 Contingency table of blocks evaluated on the GROTOAP dataset. Columns correspond to our labels, rows correspond to GROTOAP labels. Boldface entries denote equivalent labellings.

	Decoration	*Caption*	*Main*	*Heading*	*Table*	*Sparse*	unlabelled
abstract	0	0	6	1	0	50	25
body	6	25	**3707**	**1188**[*]	8	377	278
keywords	0	0	2	1	0	6	17
correspondence	0	0	0	0	0	9	4
figure_caption	0	**437**	0	0	2	106	45
table_caption	2	**167**	0	0	8	14	5
equation_label	10	0	0	0	0	183	0
page_number	**819**	0	0	0	0	26	0
unknown	3	3	76	66	345	366	377
table	14	1	2	4	**3700**	1790	46
copyright	5	0	7	0	0	75	90
type	1	0	0	0	0	101	1
author	0	0	0	2	0	4	3
editor	0	0	0	0	0	18	1
references	2	0	**379**[†]	37	0	375	516
title	0	0	0	1	0	2	6
figure	2	1	0	0	7	3706	89
dates	0	0	0	0	0	7	64
equation	0	0	2	32	0	491	3
bib_info	**1158**	0	2	3	0	115	59
affiliation	0	0	5	4	0	48	44
Total	2022	634	4188	1339	4070	7869	1673

[*] In the ground truth dataset, headings are part of the body zone of the following paragraph.
[†] We do not yet detect special reference blocks.

Table 4 Characteristics of the PubMed dataset consisting of 1000 randomly selected documents. The left table shows the diverse distribution of article types as specified by the provided metadata, the right table lists the most frequently occurring journal titles (in total there were 220 unique journals).

Article type	Count
Research article	759
Review article	75
Book review	45
Abstract	15
News	13
Editorial	13
Case report	12
Product review	11
Letter	9
Meeting report	7
Brief report	7
Correction	6
Commentary	4
Other	24

Journal title	Count
Environmental Health Perspectives	69
The Yale Journal of Biology and Medicine	38
Medical History	28
PLoS ONE	24
Nucleic Acids Research	23
BMC Bioinformatics	16
BMC Public Health	16
BMC Genomics	15
The Ulster Medical Journal	15

deletions (but not of substitutions) necessary to transform the extracted text into the actual text. Given these numbers we define precision and recall for the main text as

$$P_{text} = 1 - \frac{D}{\max(N,M)} \quad \text{and} \quad R_{text} = 1 - \frac{I}{\max(N,M)}, \quad (4)$$

respectively, where D and I are the number of deletions and insertions and N and M are the lengths of the two strings. Intuitively, a low number of deletions means that most of the extracted text is contained in the true body text in the right order, thus having a high precision. Analogously, if the number of insertions is small most of the true body text is extracted, leading to a high recall. This evaluation not only depends on the correct labelling of main text and heading blocks, but also the correct reading order, as shuffling text pieces results in reduced precision and recall values.

Since the dataset contains a range of different article types, including book reviews, abstracts, and product presentations, we included only those documents into the analysis which contain a body text and at least one section header. It can be seen in Table 5 that most of the main text is extracted correctly. Insertions (decreased recall) typically occur when main text blocks are miscategorized as e.g., captions or sparse blocks. A typical case for deletions (decreased precision) is that parts of the reference section get included into the main text although they are not part of the ground truth text. We also evaluated the effect of resolving hyphenations ("raw" in Table 5) and found that it is below 1% in precision and obviously does not affect recall.

For the evaluation of the extracted headings we collect the texts from the blocks labelled as heading and compute standard precision and recall of the ground truth headings. In the PubMed dataset headings are contained within the *title* tag of a *sec* section of the ground truth XML. Table 5 shows that performance values are around 80%. One source of error here is that the ground truth does not distinguish between normal section headings and paragraph headings, which we do not extract since they are not offset from, but part of the following main text block. In [9] performance values of over 90% are reported for heading detection, but their method is based on a combination of classification and clustering and is evaluated on a dataset of books which might be less diverse than our set of scholarly articles.

We compared our performance to a state-of-the-art system for logical structure detection, SectLabel [20] from the ParsCit package[12]. This system takes a raw text file as input and uses a trained CRF model to classify individual lines into different categories, in particular *bodyText*, *sectionHeader*, *subsectionHeader*, and *subsubsectionHeader*. We applied SectLabel on the output of two standard pdf-to-text tools, PDFBox and Poppler[13], and evaluated the extracted main text and section headings on the same subset of the PubMed dataset. Table 5 shows the performance values obtained on both the main text and the heading extraction. On the main text, which is obtained by concatenating the contents of the *bodyText* tags, a reasonable recall is achieved, however, typically more than the actual body text is categorized as such. The performance on headings is substantially lower.

For these comparisons we used the off-the-shelf model of the SectLabel system that is provided as part of their software distribution, following the practice of other researchers [3,2,11]. This model is trained on a number of scientific papers from the field of computer science that use different style guidelines [20].

Table 5 Performance of main text and heading extraction compared to the output of ParsCit (SectLabel) evaluated on a random subset of 1000 documents from the PubMed dataset. Main text performance is defined in terms of the relative number of insert and delete operations necessary to reproduce the ground truth text. "Raw" indicates performance without hyphenation resolution. ParsCit requires raw text as input, generated by PDFBox and Poppler.

Body text:

		Micro-	
	Precision	**Recall**	**F1**
Main text	**0.873**	**0.969**	**0.918**
Main text (raw)	0.871	**0.969**	0.917
ParsCit (PDFBox)	0.741	0.963	0.838
ParsCit (Poppler)	0.711	0.926	0.804
		Macro-	
	Precision	**Recall**	**F1**
Main text	**0.950**	**0.961**	**0.945**
Main text (raw)	0.947	**0.961**	0.944
ParsCit (PDFBox)	0.787	0.960	0.857
ParsCit (Poppler)	0.757	0.925	0.827

Headings:

		Micro-	
	Precision	**Recall**	**F1**
Headings	**0.748**	**0.771**	**0.760**
ParsCit (PDFBox)	0.403	0.227	0.290
ParsCit (Poppler)	0.417	0.219	0.287
		Macro-	
	Precision	**Recall**	**F1**
Headings	**0.837**	**0.768**	**0.779**
ParsCit (PDFBox)	0.392	0.269	0.299
ParsCit (Poppler)	0.421	0.259	0.300

7.4 Table of contents.

For the table of contents evaluation we use the same dataset as for the evaluation of the heading labelling. We filter the test articles from the PubMed dataset to contain only documents with available section information and no duplicate heading names, resulting in 633 documents. To measure the quality of the TOC extraction we compute the minimal tree edit distance in comparison to the heading tree from PubMed, calculated by the Zhang-Shasha algorithm[14] [31]. A distance of zero indicates that the algorithm exactly recreated the TOC tree.

Errors in the TOC extraction might originate in

i) the block extraction stage (i.e., blocks which do not exactly contain the heading text),
ii) the block categorization stage (i.e., heading blocks which are not labelled as such), or
iii) in errors introduced by the TOC extraction itself.

In the evaluation we allow up to four extra characters at the front (or back) of the extracted heading, as for example the heading numbering is sometimes not part of the ground truth. In Table 6 the achieved performance of our approach is compared to the ParsCit algorithm using its default settings. We report three runs for our system to demonstrate the impact of the different types of errors. For about half of the documents (305) all heading blocks have been correctly extracted. Here our TOC extraction algorithm produces results very close to the ground truth with an average edit distance of considerably less than 1 (the edit distance was 0 for about 89% of these articles). Including errors from the block categorization stage raises the average edit distance by about 2, and errors introduced by the block extraction stage again add roughly the same amount.

As a comparison we reconstructed the table of contents from the headings extracted by the SectLabel/ParsCit system applied the initial 633 documents. Even with all sources of errors considered, the performance of our system is considerably better than using the ParsCit approach.

[12] http://wing.comp.nus.edu.sg/parsCit/
[13] http://poppler.freedesktop.org/
[14] https://github.com/timtadh/zhang-shasha

Table 6 Performance of the table of contents extraction on the PubMed dataset measured as the average tree edit distance. Different scenarios highlight the influence of different types of errors (see text). The best performance is achieved when errors introduced by the block extraction and the categorization stages are removed. The results are compared to the TOC created from the headings extracted by ParsCit (SectLabel) applied to the text output of PDFBox and Poppler.

	TOC ext.	**TOC ext. Block cat.**	**TOC ext. Block cat. Block ext.**
error types involved	iii)	ii) & iii)	i) - iii)
Mean tree edit distance	0.25	2.15	5.18
Number of articles	308	308	633
	ParsCit (Poppler)	**ParsCit (PDFBox)**	
Mean tree edit distance	13.59	13.42	
Number of articles	633	633	

Previous work on ToC extraction focused on parsing the dedicated table of contents section of a book [5,9], which is a different problem than reconstructing it from the actual content. Furthermore, the layout of of a scientific article is much denser than that of a book, thus making the ToC extraction more difficult. The book structure extraction competition held at ICDAR 2013 [6] also targeted the reconstruction of the table of contents of digitised books. However, only few participants investigated the full content, apart from the printed ToC, and the relatively low performance values emphasize that this is a rather hard problem. To the best of our knowledge, our approach is the first that tries to reconstruct the table of contents from the full content of scientific articles.

8 Discussion

We have developed an unsupervised processing pipeline for the analysis of the structure of a scientific article given as a PDF file. Starting from the raw character stream provided by the open-source PDFBox library, we first extract contiguous text blocks that serve as the basic physical building blocks of the document. In the logical layout analysis these blocks are then ordered by the reading order and categorised into different document parts. Finally, we use this information to extract the body text and the table of contents of the given article.

All stages of our pipeline make prominent use of heuristics and unsupervised machine learning techniques. In particular, clustering algorithms such as k-means and HAC are used in the block extraction stage, the detection of the main text, and the creation of the table of contents tree. Unsupervised techniques are especially interesting because of their flexibility to adapt to new input statistics without the need to retrain a model. This could be useful in many digital library scenarios due to the changing and dynamic nature of corpora.

One major problem with PDFBox and other tools is that the information provided about individual characters in the PDF is inherently noisy, for example, height and width information might be wrong, or information about the font of some characters might be missing. This implicit noise affects every stage of our system, and we believe that its performance could be considerably improved if this low-level information would be more reliable. This has been observed by other researchers [18,17], which often use solely geometric information.

For the evaluation of our system we selected a random subset of the PubMed database, which consists of a variety of different document types and formats. This diversity is representative for the domain of scientific articles and demonstrates the generalization of our techniques. Our results shows that the performance of our system, which makes use of the formatting and layout of the article, is considerably better than the SectLabel algorithm from the ParsCit system, which operates on plain text only and which we plugged in with the off-the-shelf CRF model. It has already been shown in [20] that the inclusion of rich document features would significantly improve the detection of the logical structure. Another reason for the large performance deterioration could be that the statistics of PubMed documents are substantially different from those documents for which the SectLabel system was trained. A similar observation was made in [26], where this system is also discussed.

In future work we plan to extend our system with the detection of further document parts, such as algorithms, figure regions, footnotes, or equations. We have already achieved good results on reference extraction [14], where we detected the reference section within a paper, segmented it into individual citations, and classified the individual tokens of the reference string into author, title, journal, etc. The inclusion of formatting and layout information allowed us to improve the performance over existing approaches. Another interesting problem that we would like to tackle in the future is the parsing of tables by recreating the tabular structure [30]. The extraction of relations or facts contained therein is of particular interest for information retrieval scenarios. Finally, we believe that a comparison of our unsupervised pipeline to a fully supervised classification model would gain further valuable insight into the problem of document structure analysis.

Acknowledgements The presented work was in part developed within the CODE project funded by the EU FP7 (grant no. 296150) and the TEAM IAPP project (grant no. 251514) within the FP7 People Programme. The Know-Center is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry of Economy, Family and Youth and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

1. Aiello, M., Monz, C., Todoran, L., Worring, M.: Document understanding for a broad class of documents. International Journal on Document Analysis and Recognition **5**(1), 1–16 (2002). DOI 10.1007/s10032-002-0080-x
2. Beel, J., Langer, S., Genzmehr, M., Müller, C.: Docear's PDF Inspector: Title Extraction from PDF files. In: Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2013) (2013)
3. Constantin, A., Pettifer, S., Voronkov, A.: PDFX: Fully-automated PDF-to-XML Conversion of Scientific Literature. In: Proceedings of the 13th ACM symposium on Document Engineering (2013)
4. Councill, I.G., Giles, C.L., Kan, M.y.: ParsCit: An open-source CRF Reference String Parsing Package. In: N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odjik, S. Piperidis, D. Tapias (eds.) Proceedings of LREC, vol. 2008, pp. 661–667. Citeseer, European Language Resources Association (ELRA) (2008). DOI 10.1.1.150.6790
5. Dejean, H., Meunier, J.L.: A system for converting PDF documents into structured XML format. In: Document Analysis Systems VII, pp. 129–140 (2006)
6. Doucet, A., Kazai, G., Colutto, S., Mühlberger, G.: Overview of the ICDAR 2013 Competition on Book Structure Extraction. In: Proceedings of the Twelfth International Conference on Document Analysis and Recognition (ICDAR'2013), p. 6. Washington DC, USA (2013)
7. Esposito, F., Ferilli, S., Basile, T.M.A.: Machine Learning for Digital Document Processing : From Layout Analysis To Metadata Extraction. World Wide Web Internet And Web Information Systems **138**(2008), 1–35 (2008). DOI 10.1007/978-3-540-76280-5_5
8. Ferilli, S., Basile, T., Mauro, N.D.: Markov logic networks for document layout correction. In: Modern Approaches in Applied Intelligence, pp. 275–284 (2011)
9. Gao, L., Tang, Z., Lin, X., Liu, Y., Qiu, R., Wang, Y.: Structure extraction from PDF-based book documents. In: Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries, pp. 11–20 (2011)
10. Gorman, L.O., Definitions, A.: The Document Spectrum for Page Layout Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **15**(11), 1162–1173 (1993)
11. Granitzer, M., Hristakeva, M., Knight, R., Jack, K.: A Comparison of Metadata Extraction Techniques for Crowdsourced Bibliographic Metadata Management. In: Proceedings of the 27th Symposium On Applied Computing, p. to appear. ACM New York, NY, USA (2012)
12. Granitzer, M., Hristakeva, M., Knight, R., Jack, K., Kern, R.: A Comparison of Layout based Bibliographic Metadata Extraction Techniques. In: WIMS12 - International Conference on Web Intelligence, Mining and Semantics, pp. 19:1–19:8. ACM New York, NY, USA (2012)
13. Kern, R., Jack, K., Hristakeva, M., Granitzer, M.: TeamBeam - Meta-Data Extraction from Scientific Literature. In: 1st International Workshop on Mining Scientific Publications (2012)
14. Kern, R., Klampfl, S.: Extraction of references using layout and formatting information from scientific articles. D-Lib Magazine **19**(9/10) (2013). DOI 10.1045/september2013-kern
15. Klink, S., Dengel, A., Kieninger, T.: Document Structure Analysis Based on Layout and Textual Features. In: Proc. of International Workshop on Document Analysis Systems (2000)
16. Lin, X.: Header and Footer Extraction by Page-Association. Proceedings of SPIE **5010**, 164–171 (2002). DOI 10.1117/12.472833
17. Liu, Y., Bai, K., Mitra, P., Giles, C.L.: Improving the Table Boundary Detection in PDFs by Fixing the Sequence Error of the Sparse Lines. 2009 10th International Conference on Document Analysis and Recognition pp. 1006–1010 (2009). DOI 10.1109/ICDAR.2009.138
18. Liu, Y., Mitra, P., Giles, C.L.: A Fast Preprocessing Method for Table Boundary Detection: Narrowing Down the Sparse Lines Using Solely Coordinate Information. In: 2008 The Eighth IAPR International Workshop on Document Analysis Systems, pp. 431–438. Ieee (2008). DOI 10.1109/DAS.2008.77
19. Liu, Y., Mitra, P., Giles, C.L.: Identifying table boundaries in digital documents via sparse line detection. In: Proceeding of the 17th ACM conference on Information and knowledge mining CIKM 08, pp. 1311–1320. ACM Press (2008). DOI 10.1145/1458082.1458255
20. Luong, M.T., Nguyen, T.D., Kan, M.Y.: Logical structure recovery in scholarly articles with rich document features. International Journal of Digital Library Systems **1**(4), 1–23 (2011). DOI 10.4018/jdls.2010100101
21. Malerba, D., Ceci, M., Berardi, M.: Machine learning for reading order detection in document image understanding. Machine Learning in Document Analysis pp. 45–69 (2008)
22. Mao, S., Rosenfeld, A., Kanungo, T.: Document structure analysis algorithms: a literature survey. Proceedings of SPIE **5010**(1), 197–207 (2003). DOI 10.1117/12.476326
23. Meunier, J.L.: Optimized XY-Cut for Determining a Page Reading Order. Eighth International Conference on Document Analysis and Recognition ICDAR05 **1**, 347–351 (2005). DOI 10.1109/ICDAR.2005.182
24. Nagy, G., Seth, S., Viswanathan, M.: A prototype document image analysis system for technical journals. Computer **25**(7), 10–22 (1992). DOI 10.1109/2.144436
25. Peng, F., McCallum, A.: Accurate Information Extraction from Research Papers using Conditional Random Fields. In: HLT-NAACL04, vol. 2004, pp. 329–336 (2004). DOI 10.1.1.10.5644
26. Ramakrishnan, C., Patnia, A., Hovy, E., Burns, G.A.: Layout-Aware Text Extraction from Full-text PDF of Scientific Articles. Source code for biology and medicine **7**(1), 7 (2012). DOI 10.1186/1751-0473-7-7
27. Summers, K.: Automatic discovery of logical document structure. Ph.D. thesis (1998)
28. Tkaczyk, D., Bolikowski, L., Czeczko, A., Rusek, K.: A Modular Metadata Extraction System for Born-Digital Articles. 2012 10th IAPR International Workshop on Document Analysis Systems pp. 11–16 (2012). DOI 10.1109/DAS.2012.4
29. Tkaczyk, D., Czeczko, A., Rusek, K.: GROTOAP: ground truth for open access publications. Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries pp. 381–382 (2012)
30. Zanibbi, R., Blostein, D., Cordy, J.R.: A survey of table recognition. Document Analysis and Recognition **7**(1), 1–16 (2004). DOI 10.1007/s10032-004-0120-9
31. Zhang, K., Shasha, D.: Simple Fast Algorithms for the Editing Distance between Trees and Related Problems. SIAM Journal on Computing **18**(6), 1245–1262 (1989). DOI 10.1137/0218082