An Unsupervised Machine Learning Approach to Body Text and Table of Contents Extraction from Digital Scientific Articles

Stefan Klampfl[1], Roman Kern[1,2]

[1]Know-Center GmbH
[2] Knowledge Technologies Institute, Graz University of Technology
Graz, Austria
{sklampfl,rkern}@know-center.at, rkern@tugraz.at

Abstract. Scientific articles are predominantly stored in digital document formats, which are optimised for presentation, but lack structural information. This poses challenges to access the documents' content, for example for information retrieval. We have developed a processing pipeline that makes use of unsupervised machine learning techniques and heuristics to detect the logical structure of a PDF document. Our system uses only information available from the current document and does not require any pre-trained model. Starting from a set of contiguous text blocks extracted from the PDF file, we first determine geometrical relations between these blocks. These relations, together with geometrical and font information, are then used categorize the blocks into different classes. Based on this logical structure we finally extract the body text and the table of contents of a scientific article. We evaluate our pipeline on a number of datasets and compare it with state-of-the-art document structure analysis approaches.

1 Introduction

As the growth of the global volume of scientific literature reaches unprecedented levels, there is an increasing demand for automated processing systems that support both librarians and researchers in managing collections of scholarly articles. The tasks of these systems range from the extraction of meta-data of a paper to the extraction of the table of contents and the body text, as well as named entities and facts contained therein. An important prerequisite for these tasks is the analysis of the document structure, which is commonly distinguished into a physical or logical layout [1]. Even though today most articles are digitally produced as PDF files, this remains a challenging problem because most of them do not contain structural information, but only provide the rendering information of individual text fragments.

Here we describe a processing pipeline that performs logical layout analysis from a scientific article in PDF format and uses this information to extract its body text and table of contents. A demonstration of the system can be accessed online[1], and the source code is available under an open source license[2]. The physical structure is provided by

[1] http://knowminer.at:8080/code-demo/index.html
[2] https://www.knowminer.at/svn/opensource/projects/code/trunk

an open source tool[3] that builds upon the output of the PDFBox[4] library and that produces a set of contiguous text blocks. Our pipeline performs three main steps: First, two geometrical relations between text blocks are extracted: the reading order and the block neighbourhood (section 3). Second, the blocks are categorized into different logical labels based on their bounding boxes and font information (section 4). This categorization stage also makes use of the geometrical relations above. Finally, we extract the body text, consisting of the section headings and the main text, and the table of contents as a tree with the section headings as nodes (section 5).

These individual steps are solved using clustering techniques and heuristics. Apart from the categorization of meta-data blocks, which we reused from previous work [2], our system works completely unsupervised and model-free and uses only information provided by the current document. This sets our approach apart from a number of related studies that use a pre-trained supervised model (e.g., [3,4,5]). A performance comparison shows that for a set of articles from the biomedical domain we outperform the approach in [5] in our tasks (section 6).

2 Related Work

Document structure analysis has been a well-studied research problem for a quite some time (see [1] for a review). Early work approached the problem with mostly rule-based systems that operated on scanned document images or the output of OCR. With the advent of PDF as the dominating format for scientific articles, researchers began to analyze documents directly in digital form. We restrict our discussion of related work to a number of more recent articles on this topic.

A recent paper [6] 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. The authors also evaluate the main text flow, but do not detail any efforts in determining the reading order.

The authors of [7] 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 [8]. 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.

Other work targeted the extraction of certain aspects of PDF documents, such as meta-data [9] or tables [10]. A popular supervised learning method for structure analysis are Conditional Random Fields (CRFs) [3]. One example is the ParsCit system [4], which uses a combination of heuristics and CRFs for reference parsing. A related system is SectLabel [5], which builds upon the feature sets defined for ParsCit to detect

[3] available at the same SVN location

[4] http://pdfbox.apache.org/

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 6).

3 Extracting relations between text blocks

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 neighbourhood relation.

Reading order. The reading order on a given page is defined as a specific permutation of all text blocks on that page. We follow the approach of Aiello et al. [11], who 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 [11] for the exact definition). In addition to [11], 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 (see Fig. 1A).

More sophisticated methods for reading order detection have been defined [7,12], 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.

Block neighborhood. We employ a simple straightforward algorithm that searches for the nearest neighbour of each block on the page in each of the four main directions, viz., top, bottom, left, and right. This yields a directed neighbourhood graph of blocks on the page, since this relation is not necessarily symmetric (see Fig. 1B).

For our setup, we found this neighbourhood relation more usable than using a Voronoi diagram computed for the centres of the blocks (as in [11]) 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.

4 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. Apart from the meta-data detectors they

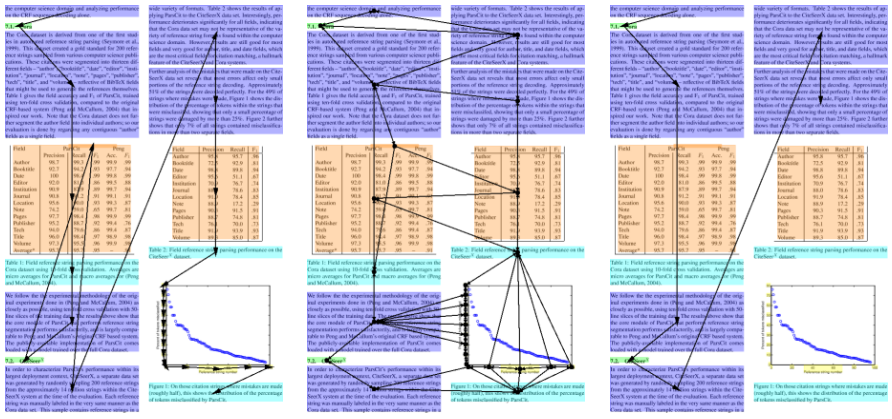

Fig. 1. Sample snapshots of our processing pipeline. All panels show the same document page with the extracted text blocks categorized into different classes (denoted by different colors). The reading order (A) and block neighbourhood (B) is determined first for all blocks on the page, later the reading order is post-processed to contain only the body text (C).

are 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.

Meta-Data. To detect meta-data blocks, which contain information about the published article, e.g., the title, the journal, or the abstract, we reused previously published work [2] which uses sequence classifiers to detect the following types of meta-data blocks: *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 [2].

Decorations. Decoration blocks contain information such as authors, titles, and page numbers and occur repeatedly at the border of each page, mostly inside headers and footers, but sometimes also at the left or right edge of the page. We adopt the work in [8], which is based on associating top and bottom lines across neighbouring pages based on both their content and their position on the page. This is considered one of the standard header/footer detection methods [7].

We use a slightly modified variant of the approach in [8] that is applicable to text blocks instead of lines. For each page, we sort all blocks on the page in four different orders: from top to bottom, from bottom to top, from left to right, and from right to left. In each ordering we consider the first 5 blocks, and for each block we calculate the maximum similarity to the 5 blocks on both the next and previous page. 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. 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 (here, 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.

Captions. Captions are text blocks usually located directly above or below a figure or table explaining its contents. To detect a *Caption* block we simply check whether its first word equals one of certain predefined keyword (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 [10,7].

Main text. 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.

HAC merges blocks bottom-up with decreasing similarity and stops once a distance threshold is reached. We chose 10 as the distance threshold, however, this parameter is not too critical; 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. 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 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, 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.

Headings. The heading detection is based on the previous labelling of main text blocks and uses additional information provided by the block relations, reading order and block neighbourhood. 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 neighbour of a main text block. Furthermore, a candidate heading block has to be either left- or centre-aligned to the following main text block. 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 neighbouring main text block, v) the distance to the neighbouring 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.

Sparse blocks and tables. The remaining blocks contain various document elements such as references, footnotes, formulas, and texts in figures and tables. According to [10] we labelled these blocks as *Sparse* blocks if (1) their width was smaller than 2/3 of the average width of a main text block, or (2) there exists a gap between two consecutive words in the block that is larger than than two times the average width between two words in the main text. Starting from a table caption (recognized by checking if it starts with a keyword such as "Table") neighbouring sparse blocks (defined by the block neighbourhood) are recursively 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.

5 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.

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. 1C). 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.

Furthermore, 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.

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 heading with similar formatting, ii) order the groups according to the heading level and iii) use the sequence information within the article and the order of groups to create a hierarchy.

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, $d(c_1, c_2) = \frac{|\overline{h_{c_1}} - \overline{h_{c_2}}|}{\min(\overline{h_{c_1}}, \overline{h_{c_2}})} + 0.1\frac{|\overline{l_{c_1}} - \overline{l_{c_2}}|}{\max(\overline{l_{c_1}}, \overline{l_{c_2}})}$, 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 by the following precedence rules: i) number of prefix segments, ii) difference in mean character height, 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 a TOC tree by exploiting the sequence information. 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. If the level of the current heading is higher than that of the preceding heading it is added as a direct child of the root node. If its level is lower it is added as a sibling to the last heading of the same level. By enforcing a valid tree hierarchy this approach corrects some errors made during the first two stages.

6 Evaluation

First, we evaluate the quality of the block categorization on the GROTOAP dataset, which consists of labelled segmentations of scientific documents into zones. Second, 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.

Table 1. 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.

Block categorization. We evaluate the block categorization on the GROTOAP dataset [13]. This dataset consists of 113 documents from various open access journals in digital form as well as their geometric hierarchical structure in XML format. It provides a ground truth for both the segmentation of document pages into contiguous text blocks and their labelling. In a related paper the same authors use this dataset to evaluate their own block extraction and meta-data categorization process [14].

This dataset provides a rather fine-grained labelling of the various text blocks (see row headings in Table 1). 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 table is shown in Table 1. 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 neighbouring 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.

Table 2. 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-			Macro-		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Main text	**0.873**	**0.969**	**0.918**	**0.950**	**0.961**	**0.945**
Main text (raw)	0.871	**0.969**	0.917	0.947	**0.961**	0.944
ParsCit (PDFBox)	0.741	0.963	0.838	0.787	0.960	0.857
ParsCit (Poppler)	0.711	0.926	0.804	0.757	0.925	0.827

Headings:

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

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.

Body text and headings. We use a dataset of 1000 randomly selected documents from PubMed[5], 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.

We evaluate the quality of the extracted main text (the content of heading and main text blocks in the reading order, see section 5) 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 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 $P_{text} = 1 - D/\max(N, M)$ and $R_{text} = 1 - I/\max(N, M)$, 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 in-

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

sertions 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 2 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 2) and found that it is below 1% in precision and obviously does not affect recall.

In the PubMed dataset headings are contained within the *title* tag of a *sec* section. For the evaluation of the extracted headings we collect the texts from the blocks labelled as heading and compute standard precision and recall. Table 2 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.

We compared our performance to a state-of-the-art system for logical structure detection, SectLabel [5] from the ParsCit package[6]. 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[7], and evaluated the extracted main text and section headings on the same subset of the PubMed dataset. Table 2 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.

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[8] [15]. 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 (e.g., blocks which do not exactly contain the heading text), ii) the block categorization stage (e.g., 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

[6] http://wing.comp.nus.edu.sg/parsCit/
[7] http://poppler.freedesktop.org/
[8] https://github.com/timtadh/zhang-shasha

Table 3. 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 ParsCit (SectLabel) output applied to the text output of PDFBox and Poppler.

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

(or back) of the extracted heading, as for example the heading numbering is sometimes not part of the ground truth. In Table 3 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. Even with all sources of errors considered, the performance of our system is considerably better than the ParsCit approach.

7 Discussion & Conclusions

We have developed an unsupervised processing pipeline that performs logical layout analysis and extracts the body text and the table of contents from a scientific article given as a PDF file, from which we extract the text stream using PDFBox. The problem with this 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.

On the other hand, our evaluation 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 [5] 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 [6], where this system is also discussed. We would like to further investigate this in future work, where we plan to compare our unsupervised pipeline to a fully supervised classification model.

Acknowledgments

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. Mao, S., Rosenfeld, A., Kanungo, T.: Document structure analysis algorithms: a literature survey. Proceedings of SPIE **5010**(1) (2003) 197–207
2. Kern, R., Jack, K., Hristakeva, M., Granitzer, M.: TeamBeam - Meta-Data Extraction from Scientific Literature. In: 1st International Workshop on Mining Scientific Publications. (2012)
3. Peng, F., McCallum, A.: Accurate Information Extraction from Research Papers using Conditional Random Fields. In: HLTNAACL04. Volume 2004. (2004) 329–336
4. Councill, I.G., Giles, C.L., Kan, M.y.: ParsCit: An open-source CRF Reference String Parsing Package. In: Proceedings of LREC. Volume 2008., Citeseer, European Language Resources Association (ELRA) (2008) 661–667
5. 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) (January 2011) 1–23
6. 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) (2012) 7
7. 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. (2011) 11–20
8. Lin, X.: Header and Footer Extraction by Page-Association. Proceedings of SPIE **5010** (2002) 164–171
9. Granitzer, M., Hristakeva, M., Knight, R., Jack, K., Kern, R.: A Comparison of Layout based Bibliographic Metadata Extraction Techniques. In: WIMS'12 - International Conference on Web Intelligence, Mining and Semantics, ACM New York, NY, USA (2012) 19:1–19:8
10. 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, ACM Press (2008) 1311–1320
11. 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) (2002) 1–16
12. Malerba, D., Ceci, M., Berardi, M.: Machine learning for reading order detection in document image understanding. Machine Learning in Document Analysis (2008) 45–69
13. 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 (2012) 381–382
14. 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 (March 2012) 11–16
15. Zhang, K., Shasha, D.: Simple Fast Algorithms for the Editing Distance between Trees and Related Problems. SIAM Journal on Computing **18**(6) (December 1989) 1245–1262