IJDAR (2015) 18:317–335
DOI 10.1007/s10032-015-0249-8

ORIGINAL PAPER

CERMINE: automatic extraction of structured metadata from scientific literature

Dominika Tkaczyk[1] · **Paweł Szostek**[1] · **Mateusz Fedoryszak**[1] · **Piotr Jan Dendek**[1] · **Łukasz Bolikowski**[1]

Received: 10 December 2014 / Revised: 4 May 2015 / Accepted: 19 June 2015 / Published online: 3 July 2015

Abstract CERMINE is a comprehensive open-source system for extracting structured metadata from scientific articles in a born-digital form. The system is based on a modular workflow, whose loosely coupled architecture allows for individual component evaluation and adjustment, enables effortless improvements and replacements of independent parts of the algorithm and facilitates future architecture expanding. The implementations of most steps are based on supervised and unsupervised machine learning techniques, which simplifies the procedure of adapting the system to new document layouts and styles. The evaluation of the extraction workflow carried out with the use of a large dataset showed good performance for most metadata types, with the average F score of 77.5 %. CERMINE system is available under an open-source licence and can be accessed at http://cermine.ceon.pl. In this paper, we outline the overall workflow architecture and provide details about individual steps implementations. We also thoroughly compare CERMINE to similar solutions, describe evaluation methodology and finally report its results.

✉ Dominika Tkaczyk
d.tkaczyk@icm.edu.pl

Paweł Szostek
pawel.szostek@gmail.com

Mateusz Fedoryszak
m.fedoryszak@icm.edu.pl

Piotr Jan Dendek
p.dendek@icm.edu.pl

Łukasz Bolikowski
l.bolikowski@icm.edu.pl

[1] Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, ul. Prosta 69, 00-838 Warsaw, Poland

Keywords Metadata extraction · Bibliography extraction · Content classification · Reference parsing · Scientific literature analysis

1 Introduction

Academic literature is a very important communication channel in the scientific world. Keeping track of the latest scientific findings and achievements, typically published in journals or conference proceedings, is a crucial aspect of the research work. Ignoring this task can result in deficiencies in the knowledge related to the latest discoveries and trends, which in turn can lower the quality of the research, make results assessment much harder and significantly limit the possibility to find new interesting research areas and challenges. Unfortunately, studying scientific literature, and in particular being up-to-date with the latest positions, is difficult and extremely time-consuming. The main reason for this is huge and constantly growing volume of scientific literature, and also the fact that publications are mostly available in the form of unstructured text.

Modern digital libraries support the process of studying the literature by providing intelligent search tools, proposing similar and related documents, building citation and author networks, and so on. In order to provide such high-quality services, the library requires an access not only to the sources of stored documents, but also to their metadata including information such as title, authors, keywords, abstract or bibliographic references. Unfortunately, in practice good quality metadata is not always available, sometimes it is missing, full of errors or fragmentary. In such cases, the library needs a reliable automatic method to extract metadata and references from documents at hand.

Even limited to analysing scientific literature only, the problem of extracting the document's metadata remains difficult and challenging, mainly due to the vast diversity of possible layouts and styles used in articles. In different documents, the same type of information can be displayed in different places using a variety of formatting styles and fonts. For instance, a random subset of 125,000 documents from PubMed Central [1] contains publications from nearly 500 different publishers, many of which use original layouts and styles in their articles. What is more, PDF format, which is currently the most popular format for storing source documents, does not preserve the information related to the document's structure, such as words and paragraphs, lists and enumerations, the structure of tables, the hierarchy of sections, or the reading order of the text. This information has to be reverse engineered based on the text content and the way the text is displayed in the source file.

These problems are addressed by CERMINE—a comprehensive tool for automatic metadata extraction from born-digital scientific literature. The extraction algorithm proposed by CERMINE performs a thorough analysis of the input scientific publication in PDF format and extracts:

- a rich set of document's metadata,
- a list of bibliographic references along with their metadata,
- structured full text with sections and subsections (currently in experimental phase).

CERMINE is based on a modular workflow composed of three paths and a number of steps with carefully defined input and output. By virtue of such workflow architecture, individual steps can be maintained separately. As a result, it is easy to perform evaluation or training, improve or replace one step implementation without changing other parts of the workflow.

Designed as a universal solution, CERMINE is able to handle a vast variety of publication layouts reasonably well, instead of being perfect in processing a limited number of document layouts only. We achieved this by employing supervised and unsupervised machine learning algorithms trained on large diverse datasets. This decision also resulted in increased maintainability of the system, as well as its ability to adapt to new, previously unseen document layouts.

The evaluation we conducted showed good performance of the key process steps and the entire metadata extraction process, with the overall F score of 77.5 % (the details are provided in Sect. 5.5). The comparison to other similar systems showed CERMINE performs better for most metadata types.

CERMINE web service, as well as the source code, can be accessed online [2].

This article is an extended version of the conference paper describing CERMINE system [3]. In contrast to the previous version, the article contains:

- detailed descriptions of all the extraction algorithm components,
- the details related to feature selection for zone classifiers,
- new evaluation results for algorithms trained on GROTOAP2 dataset [4],
- the evaluation of the bibliography extraction workflow,
- the comparison to other similar systems.

In the following sections, we describe the state of the art, provide the details about the overall workflow architecture and individual implementations and finally report the evaluation methodology and its results.

2 State of the art

Extracting metadata from articles and other documents is a well-studied problem. Older approaches expected scanned documents on the input and were prepared for executing full digitization from bitmap images. Nowadays, we have to deal with growing amount of born-digital documents, which do not require individual character recognition. The approaches to the problem differ in the scope of the solution, supported file formats and methods and algorithms used.

Most approaches focus on extracting the article's metadata only and often do not process the entire input document. Proposed solutions are usually based on rules and heuristics or machine learning techniques.

For example, Giuffrida et al. [5] extract the content from PostScript files using a tool based on `pstotext`, while basic document metadata is extracted by a set of rules and features computed for extracted text chunks. Another example of a rule-based system is PDFX described by Constatin et al. [6]. PDFX can be used for converting scholarly articles in PDF format to their XML representation by annotating fragments of the input documents and extracts basic metadata, structured full text and unparsed reference strings. Pdf-extract [7] is an open-source tool for identifying and extracting semantically significant regions of scholarly articles in PDF format. It uses a combination of visual cues and content traits to perform structural analysis in order to determine columns, headers, footers and sections, detect references sections and finally extract individual references.

Machine learning-based approaches are far more popular. They differ in classification algorithms, document fragments that undergo the classification (text chunks, lines or blocks) and extracted features. For example, Han et al. [8] extract metadata from the headers of scientific papers by two-stage classification of text lines with the use of support vector machines and text-related features. Another example of SVM-based approach is metadata extractor used in CRIS systems proposed by Kovacevic et al. [9]. The tool classifies the lines of text using both geometric and text-related features in order to extract the document's metadata from PDFs. Lu et al. [10] analyse scanned scientific journals in order to obtain volume level, issue level and article level metadata. In their approach, the pages are first OCRed, rule-based pattern matching is used for volume and issue title pages, while article metadata is extracted using SVM and both geometric and textual features of text lines.

Other classification techniques include for example hidden Markov models, neural classifiers, maximum entropy and conditional random fields. Marinai [11] extracts characters from PDF documents using JPedal package, performs rule-based page segmentation, and finally employs neural classifier for zone classification. Cui and Chen [12] use HMM classifier to extract metadata from PDF documents, while text extraction and page segmentation are done by `pdftohtml`, a third-party open-source tool. The system based on TeamBeam algorithm proposed by Kern et al. [13] is able to extract a basic set of metadata from PDF documents using an enhanced Maximum Entropy classifier. Lopez [14] proposes GROBID system for analysing scientific texts in PDF format. GROBID uses CRF in order to extract document's metadata, full text and a list of parsed bibliographic references. ParsCit, described by Luong et al. [15] also uses CRF for extracting the logical structure of scientific articles, including the document's metadata, structured full text and parsed bibliography. ParsCit analyses documents in text format, and therefore does not use geometric hints present in the PDF files.

Reference sections are typically located in the documents using heuristics [6,7,16,17] or machine learning [14,18]. Citation parsing, that is extracting metadata from citation strings, is usually performed using regular expressions and knowledge-based approaches [19,20], or more popular machine learning techniques, such as CRF [16–18,21], SVM [22] or HMM [23].

A number of systems mentioned above are available online: PDFX [24] (the tool is closed source, available only as a web service), GROBID [25], ParsCit [26] and Pdf-extract [7]. In Sect. 5.6, we report the results of comparing the performance of these tools with CERMINE. Table 1 shows the scope of the information various metadata extraction systems are able to extract.

The most important features differentiating CERMINE from other approaches are:

– CERMINE is able to extract bibliographic information related to the document, such as journal name, volume, issue or pages range.
– The algorithms use not only the text content of the document, but also its geometric features related to the way the text is displayed in the source PDF file.
– Our solution is based mostly on machine learning, which increases its ability to conform to different article layouts.
– The flexibility of the system implementation is granted by its modular architecture.
– For most metadata types, the solution is very effective.
– The source code is open and the web service is available online [2].

Table 1 The comparison of the scope of the information extracted by various metadata extraction systems

	CERMINE	PDFX	GROBID	ParsCit	Pdf-extract
Title	✓	✓	✓	✓	✓
Author	✓	✓	✓	✓	×
Affiliation	✓	×	✓	✓	×
Affiliation's metadata	✓	×	✓	×	×
Author–affiliation	✓	×	✓	×	×
Email address	✓	✓	✓	✓	×
Author–email	✓	×	✓	×	×
Abstract	✓	✓	✓	✓	×
Keywords	✓	×	✓	✓	×
Journal	✓	×	×	×	×
Volume	✓	×	×	×	×
Issue	✓	×	×	×	×
Pages range	✓	×	×	×	×
Year	✓	×	✓	×	×
DOI	✓	×	✓	×	×
Reference	✓	✓	✓	✓	✓
Reference's metadata	✓	×	✓	✓	×

The table shows simple metadata types (e.g. *title*, *author*, *abstract* or *bibliographic references*), relations between them (*author–affiliation*, *author–email address*), and also metadata in the structured form (*references* and *affiliations* along with their metadata)

3 System architecture

CERMINE accepts a scientific publication in PDF format on the input. The extraction algorithm inspects the entire content of the document and produces two kinds of output: the document's metadata and bibliography.

CERMINE's extraction workflow is composed of three paths (Fig. 1):

(A) Basic structure extraction path takes a PDF file on the input and produces its geometric hierarchical representation, which stores the entire text content of the input document and the geometric features related to the way the text is displayed in the PDF file. More precisely, the structure is composed of pages, zones, lines, words and characters, along with their coordinates and dimensions. Additionally, the reading order of all elements is set and every zone is labelled with one of four general categories: *metadata*, *references*, *body* or *other*.

(B) Metadata extraction path analyses *metadata* parts of the geometric hierarchical structure and extracts a rich set of document's metadata from them.

(C) Bibliography extraction path analyses parts of the structure labelled as *references*. The result is a list of document's parsed bibliographic references.

Table 2 shows the decomposition of the extraction workflow into paths and steps and provides basic information about tools and algorithms used for every step.

3.1 Models and formats

CERMINE's input document format is PDF, currently the most popular format for storing the sources of scientific publications. A PDF file contains by design the text of the document in the form of a list of chunks of various length specifying the position, size and other geometric features of the text as well as the information related to the fonts and graphics. PDF documents look the same no matter what software or hardware is used for viewing them. Unfortunately, the format does not preserve any information related to the logical structure of the text, such as words, lines, paragraphs, enumerations, sections, section titles or even the reading order of text chunks. This information has to be deduced from the geometric features of the text.

Currently, the extraction workflow does not include any OCR phase, it analyses only the PDF text stream found in the input document. As a result, PDF documents containing scanned pages in the form of images will not be properly processed. We plan to provide this functionality in the future. Thanks to the flexible architecture of the workflow, the only required change is adding an alternative implementation of the character extraction step, able to perform optical character recognition on scanned pages and extract characters along with dimensions and positions. Other parts of the workflow will remain the same.

CERMINE's intermediate model of the document constructed during the first process path is a hierarchical structure that holds the entire text content of the article, while also preserving the information related to the way elements are displayed in the corresponding PDF file. In this representation, an article is a list of pages, each page contains a list of zones, each zone contains a list of lines, each line contains a list of words, and finally each word contains a list of characters. Each structure element can be described by its text content and bounding box (a rectangle enclosing the element). The structure contains also the natural reading order for the elements on each level. Additionally, labels describing the role in the document are assigned to zones.

The smallest elements in the structure are individual characters. A word is a continuous sequence of characters placed in one line with no spaces between them. Punctuation marks and typographical symbols can be separate words or parts of adjacent words, depending on the presence ofspaces.

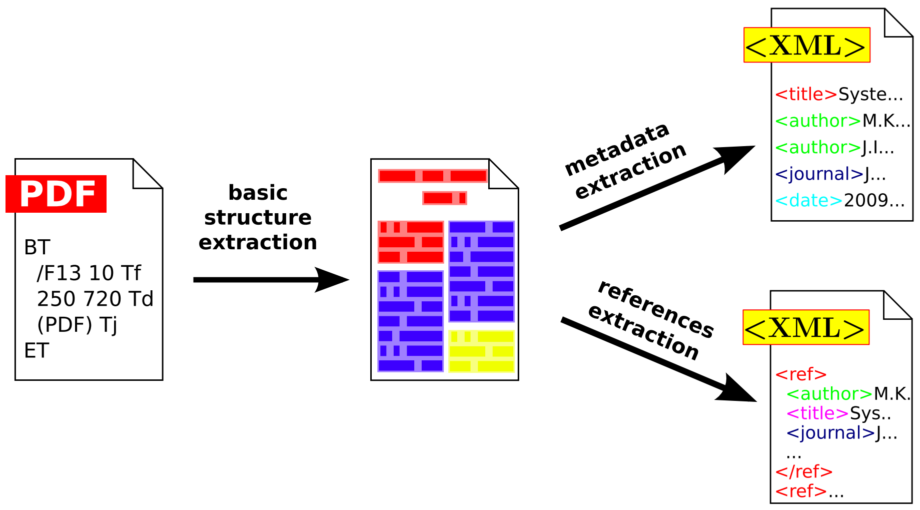


Fig. 1 CERMINE's extraction workflow architecture. At the beginning, the basic structure is extracted from the PDF file. Then, metadata and bibliography are extracted in two parallel paths

Table 2 The decomposition of CERMINE's extraction workflow into independent processing paths and steps

Path	Step	Goal	Implementation
A. Basic structure extraction	A1. Character extraction	Extracting individual characters along with their page coordinates and dimensions from the input PDF file	iText library
	A2. Page segmentation	Constructing the document's geometric hierarchical structure containing (from the top level) pages, zones, lines, words and characters, along with their page coordinates and dimensions	Enhanced Docstrum
	A3. Reading order resolving	Determining the reading order for all structure elements	Bottom-up heuristic-based
	A4. Initial zone classification	Classifying the document's zones into four main categories: *metadata*, *body*, *references* and *other*	SVM
B. Metadata extraction	B1. Metadata zone classification	Classifying the document's zones into specific metadata classes	SVM
	B2. Metadata extraction	Extracting atomic metadata information from labelled zones	Simple rule-based
C. Bibliography extraction	C1. Reference strings extraction	Dividing the content of *references* zones into individual reference strings	K-means clustering
	C2. Reference parsing	Extracting metadata information from references strings	CRF

Hyphenated words that are divided into two lines appear in the structure as two separate words that belong to different lines. A line is a sequence of words that forms a consistent fragment of the document's text. Words placed geometrically in the same line of the page, that are parts of neighbouring columns, in the structure do not belong to the same line. A zone is a consistent fragment of the document's text, geometrically separated from surrounding fragments and not divided into paragraphs or columns.

All bounding boxes are rectangles with edges parallel to the page's edges. A bounding box is defined by two points: left upper corner and right lower corner of the rectangle. The coordinates are given in typographic points (1 typographic point equals to 1/72 of an inch). The origin of the coordinate system is the left upper corner of the page.

The model can be serialized using XML TrueViz format [27]. The listing below shows a fragment of an example TrueViz file. Repeated fragments or fragments that are not used by the system have been omitted.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Document SYSTEM "Trueviz.dtd">
<Document>
 [...]
 <Page>
   <PageID Value="0"/>
   [...]
   <PageNext Value="1"/>
   <Zone>
     <ZoneID Value="0"/>
     <ZoneCorners>
       <Vertex x="55.4" y="34.3"/>
       <Vertex x="250.5" y="58.3"/>
     </ZoneCorners>
     <ZoneNext Value="1"/>
     <Classification>
       <Category Value="BIB_INFO"/>
       <Type Value=""/>
     </Classification>
     <Line>
       <LineID Value="0"/>
       <LineCorners>
         <Vertex x="55.4" y="34.3"/>
         <Vertex x="250.5" y="58.3"/>
       </LineCorners>
       <LineNext Value="1"/>
       <LineNumChars Value=""/>
       <Word>
         <WordID Value="0"/>
         <WordCorners>
           <Vertex x="55.4" y="34.3"/>
           <Vertex x="115.3" y="58.3"/>
         </WordCorners>
         <WordNext Value="1"/>
         <WordNumChars Value=""/>
         <Character>
           <CharacterID Value="0"/>
           <CharacterCorners>
             <Vertex x="55.4" y="34.3"/>
             <Vertex x="74.1" y="58.3"/>
           </CharacterCorners>
           <CharacterNext Value="1"/>
           <GT_Text Value="B"/>
         </Character>
         <Character>
         [...]
       </Word>
       [...]
     </Line>
     [...]
   </Zone>
 </Page>
</Document>
```

The output format of the extraction workflow is NLM JATS [28]. JATS (Journal Article Tag Suite) defines a rich set of XML elements and attributes for describing scientific publications and is an application of NISO Z39.96-2012 standard [29]. Documents in JATS format can store a wide range of structured metadata of the document (title, authors, affiliations, abstract, journal name, identifiers, etc.), the full text (the hierarchy of sections, headers and paragraphs, structured tables, equations, etc.), the document's bibliography in the form of a list of references along with their identifiers and metadata, and also the information related to the text formatting.

4 Extraction workflow implementation

In this section, we describe in detail the approaches and algorithms used to implement all the individual workflow steps.

4.1 Layout analysis

Layout analysis is the initial phase of the entire workflow. Its goal is to create a hierarchical structure of the document preserving the entire text content of the input document and features related to the way the text is displayed in the PDF file.

Layout analysis is composed of the following steps:

1. Character extraction (A1)—extracting individual characters from a PDF document.
2. Page segmentation (A2)—joining characters into words, lines and zones.
3. Reading order determination (A3)—calculating the reading order for all the structure levels.

4.1.1 Character extraction

The purpose of the character extraction step is to extract individual characters from the PDF stream along with their positions on the page, widths and heights. These geometric parameters play important role in further steps, in particular page segmentation and content classification.

The implementation of character extraction is based on open-source iText [30] library. We use iText to iterate over PDF's text-showing operators. During the iteration, we extract text strings along with their size and position on the page. Next, extracted strings are split into individual characters and their individual widths and positions are calculated. The result is an initial flat structure of the document, which consists only of pages and characters. The widths and heights computed for individual characters are approximate and can slightly differ from the exact values depending on the font, style and characters used. Fortunately, those approximate values are sufficient for further steps.

4.1.2 Page segmentation

The goal of page segmentation step is to create a geometric hierarchical structure storing the document's content. As a result the document is represented by a list of pages, each page contains a set of zones, each zone contains a set of text lines, each line contains a set of words, and finally each word contains a set of individual characters. Each object in the structure has its content, position and dimensions. The structure is heavily used in further steps, especially zone classification and bibliography extraction.

Page segmentation is implemented with the use of a bottom-up Docstrum algorithm [31]:

1. The algorithm is based to a great extent on the analysis of the nearest-neighbour pairs of individual characters. In the first step, five nearest components for every character are identified (red lines in Fig. 2).
2. In order to calculate the text orientation (the skew angle), we analyse the histogram of the angles between the elements of all nearest-neighbour pairs. The peak value is assumed to be the angle of the text. Since in the case of born-digital documents, the skew is almost always horizontal, and this step is mostly useful for documents containing scanned pages.
3. Next, within-line spacing is estimated by detecting the peak of the histogram of distances between the nearest neighbours. For this histogram, we use only those pairs, in which the angle between components is similar to the estimated text orientation angle (blue lines in Fig. 2). All the histograms used in Docstrum are smoothed to avoid detecting local abnormalities. An example of a smoothed distance histogram is shown in Fig. 3.
4. Similarly, between-line spacing is also estimated with the use of a histogram of the distances between the nearest-neighbour pairs. In this case, we include only those pairs that are placed approximately in the line perpendicular to the text line orientation (green lines in Fig. 2).
5. Next, line segments are found by performing a transitive closure on within-line nearest-neighbour pairs.

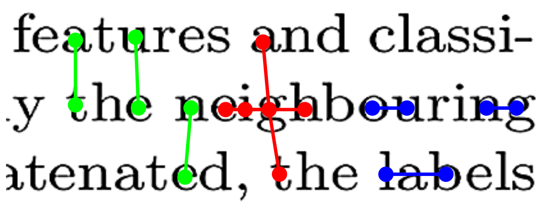


Fig. 2 An example fragment of a text zone in a scientific article. The figure shows five nearest neighbours of a given character (*red lines*), neighbours placed in the same line used to determine in-line spacing (*blue lines*), and neighbours placed approximately in the line perpendicular to the text line orientation used to determine between-line spacing (*green lines*) (color figure online)

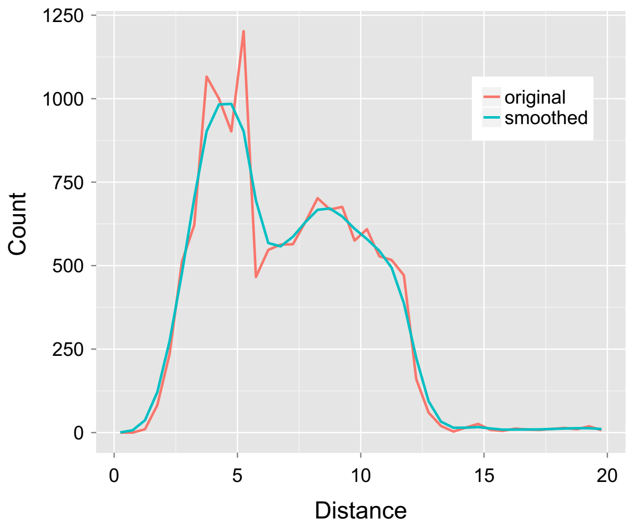


Fig. 3 An example of a nearest-neighbour distance histogram. The figure shows both original and smoothed versions of the histogram. The peak distance chosen based on the original data would be the global maximum, even though the histogram contains two close peaks of similarly high frequency. Thanks to smoothing both local peaks are taken into account, shifting the resulting peak slightly to the left and yielding more reliable results

To prevent joining line segments belonging to different columns, the components are connected only if the distance between them is sufficiently small.

6. The zones are then constructed by grouping the line segments on the basis of heuristics related to spatial and geometric characteristics: parallelness, distance and overlap.
7. The segments belonging to the same zone and placed in one line horizontally are merged into final text lines.
8. Finally, we divide the content of each text line into words based on within-line spacing.

A few improvements were added to the Docstrum-based implementation of page segmentation:

– the distance between connected components, which is used for grouping components into lines, has been split into horizontal and vertical distance (based on estimated text orientation angle),
– fixed maximum distance between lines that belong to the same zone has been replaced with a value scaled relatively to the line height,
– merging of lines belonging to the same zone has been added,
– rectangular smoothing window has been replaced with Gaussian smoothing window,
– merging of highly overlapping zones has been added,
– words determination based on within-line spacing has been added.

4.1.3 Reading order resolving

A PDF file contains by design a stream of strings that undergoes extraction and segmentation process. As a result, we obtain pages containing characters grouped into zones, lines and words, all of which have a form of unsorted bag of items. The aim of setting the reading order is to determine the right sequence in which all the structure elements should be read. This information is used in zone classifiers and also allows to extract the full text of the document in the right order. An example document page with a reading order of the zones is shown in Fig. 4.

Reading order resolving algorithm is based on a bottom-up strategy: first characters are sorted within words and words within lines horizontally, then lines are sorted vertically within zones, and finally we sort zones. The fundamental principle for sorting zones was taken from [32]. We make use of an observation that the natural reading order in most modern languages descends from top to bottom, if successive zones are aligned vertically, otherwise it traverses from left to right. There are few exceptions to this rule, for example, Arabic script, and such cases would not be handled properly by the algorithm. This observation is reflected in the distances counted for all zone pairs: the distance is calculated using the angle of the slope of the vector connecting

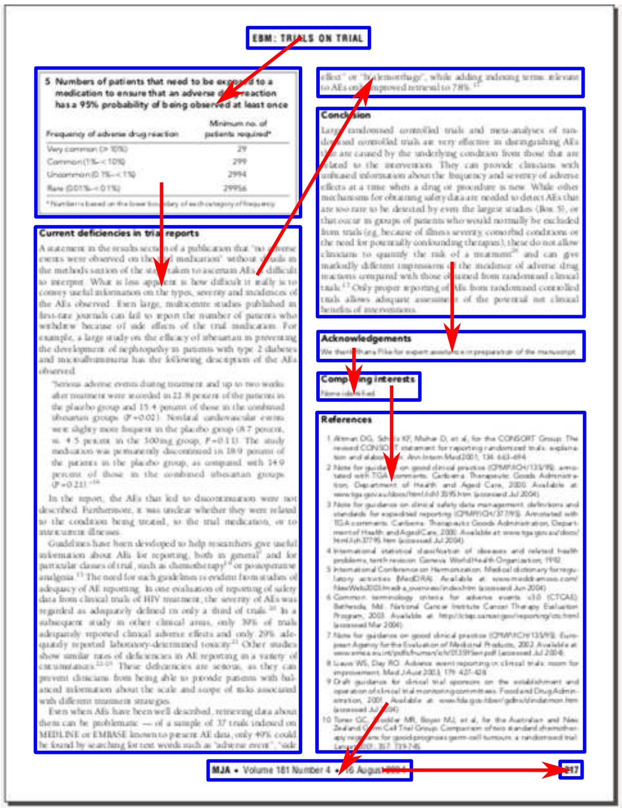

EBM: TRIALS ON TRIAL
5 Numbers of patients that need to be exposed to a medication to ensure that an adverse drug reaction has a 95% probability of being observed at least once
Current deficiencies in trial reports
Conclusion
Acknowledgements
Competing interests
References
MJA • Volume 181 Number 4 • 16 August 2004
217

Fig. 4 An example page from a scientific publication. The image shows the zones and their reading order

zones. As a result, zones aligned vertically are in general closer than those aligned horizontally. Then, using an algorithm similar to hierarchical clustering methods, we build a binary tree by repeatedly joining the closest zones and groups of zones. After that, for every node its children are swapped, if needed. Finally, an in order tree traversal gives the desired zones order.

4.2 Content classification

The goal of content classification is to determine the role played by every zone in the document. This is done in two steps: initial zone classification (A4) and metadata zone classification (B1).

The goal of initial classification is to label each zone with one of four general classes: *metadata* (document's metadata, e.g. title, authors, abstract, keywords, and so on), *references* (the bibliography section), *body* (publication's text, sections, section titles, equations, figures and tables, captions) or *other* (acknowledgments, conflicts of interests statements, page numbers, etc.).

The goal of metadata zone classification is to classify all *metadata* zones into specific metadata classes: *title* (the title of the document), *author* (the names of the authors), *affiliation* (authors' affiliations), *editor* (the names of the editors), *correspondence* (addresses and emails), *type* (the type specified in the document, such as "research article", "editorial" or "case study", *abstract* (document's abstract), *keywords* (keywords listed in the document), *bib_info* (for zones containing bibliographic information, such as journal name, volume, issue, DOI, etc.), *dates* (the dates related to the process of publishing the article).

The classifiers are implemented in a similar way. They both employ support vector machines, and the implementation is based on LibSVM library [33]. They differ in target zone labels, extracted features and SVM parameters used. The features, as well as SVM parameters were selected using the same procedure, described in Sects. 4.2.1 and 4.2.2.

Support vector machines is a very powerful classification technique able to handle a large variety of input and work effectively even with training data of a small size. The algorithm is based on finding the optimal separation hyperplane and is little prone to overfitting. It does not require a lot of parameters and can deal with highly dimensional data. SVM is widely used for content classification and achieves very good results in practice.

The decision of splitting content classification into two separate classification steps, as opposed to implementing only one zone classification step, was based mostly on aspects related to the workflow architecture and maintenance. In fact both tasks have different characteristics and needs. The goal of the initial classifier is to divide the article's content into three general areas of interest, which can be then analysed independently in parallel, while metadata classifier performs far more detailed analysis of only a small subset of all zones.

The implementation of the initial classifier is more stable: the target label set does not change, and once trained on a reasonably large and diverse dataset, the classifier performs well on other layouts as well. On the other hand, metadata zones have much more variable characteristics across different layouts, and from time to time there is a need to tune the classifier or retrain it using a wider document set. What is more, sometimes the classifier has to be extended to be able to capture new labels, not considered before (for example a special label for zones containing both author and affiliation, a separate label for categories or general terms).

For these reasons, we decided to implement content classification in two separate steps. As a result, we can maintain them independently, and for example adding another metadata label to the system does not change the performance of recognizing the bibliography sections. It is also possible that in the future the metadata classifier will be reimplemented using a different technique, allowing to add new training cases incrementally, for example using a form of online learning.

For completeness, we compared the performance of a single zone classifier assigning all needed labels in one step to the classifier containing two separate classifiers executed in a sequence (our current solution). The results can be found in Sect. 5.3.

4.2.1 Feature selection

The features used by the classifiers were selected with the use of the zone validation dataset (all the datasets used for experiments are described in Sect. 5.1). For each classifier, we analysed 97 features in total. The features capture various aspects of the content and surroundings of the zones and can be divided into the following categories:

- geometric—based on geometric attributes, some examples include: zone's height and width, height to width ratio, zone's horizontal and vertical position, the distance to the nearest zone, empty space below and above the zone, mean line height, whether the zone is placed at the top, bottom, left or right side of the page;
- lexical—based upon keywords characteristic for different parts of narration, such as: affiliations, acknowledgments, abstract, keywords, dates, references, or article type; these features typically check, whether the text of the zone contains any of the characteristic keywords;
- sequential—based on sequence-related information, some examples include the label of the previous zone (according to the reading order) and the presence of the

same text blocks on the surrounding pages, whether the zone is placed in the first/last page of the document;
- formatting—related to text formatting in the zone, examples include font size in the current and adjacent zones, the amount of blank space inside zones, mean indentation of text lines in the zone;
- heuristics—based on heuristics of various nature, such as the count and percentage of lines, words, uppercase words, characters, letters, upper/lowercase letters, digits, whitespaces, punctuation, brackets, commas, dots, etc; also whether each line starts with enumeration-like tokens, or whether the zone contains only digits.

In general, feature selection was performed by analysing the correlations between the features and between features and expected labels. For simplicity, we treat all the features as numerical variables; the values of binary features are decoded as 0 or 1. The labels, on the other hand, are an unordered categorical variable.

Let L be a set of zone labels for a given classifier, n the number of the observations (zones) in the validation dataset and $k = 97$ the initial number of analysed features. For ith feature, where $0 \leq i < k$, we can define $f_i \in R^n$, a vector of the values of the feature ith for subsequent observations. Let also $l \in L^n$ be the corresponding vector of zone labels.

In the first step, we removed redundant features, highly correlated with other features. For each pair of feature vectors, we calculated the Pearson's correlation score and identified all the pairs $f_i, f_j \in R^n$, such that

$$|corr(f_i, f_j)| > 0.9$$

Next, for every feature from highly correlated pairs, we calculated the mean absolute correlation:

$$meanCorr(f_i) = \frac{1}{k}\sum_{j=0}^{k-1} corr(f_i, f_j)$$

and from each highly correlated pair, the feature with higher $meanCorr$ was eliminated. This left us with 78 and 75 features for initial and metadata classifiers, respectively. Let's denote the number of remaining features as k'.

After eliminating features using correlations between them, we analysed the features using their associations with the expected zone labels vector l. To calculate the correlation between a single feature vector f_i (numeric) and label vector l (unordered categorical), we employed Goodman and Kruskal's τ (tau) measure [34]. Let's denote it as $\tau(f_i, l)$.

Let $f_0, f_1, \ldots f_{k'-1}$ be the sequence of the feature vectors ordered by non-decreasing τ measure, that is

$$\tau(f_0, l) \leq \tau(f_1, l) \leq \cdots \leq \tau(f_{k'-1}, l)$$

The features were then added to the classifier one by one, starting from the best one (the mostly correlated with the labels vector, $f_{k'-1}$), and at the end the classifier contained the entire feature set. At each step, we performed a fivefold cross-validation on the validation dataset and calculated the overall F score as an average for individual labels. For completeness, we also repeated the same process with reversed order of the features, starting with less useful features. The results for initial and metadata classifier are shown in Figs. 5 and 6, respectively.

Using these results, we eliminated a number of the least useful features $f_0, f_1, \ldots f_t$, such that the performance of the classifier with the remaining features was similar to the performance of the classifier trained on the entire feature set. Final feature sets contain 53 and 51 features for initial and metadata classifier, respectively.

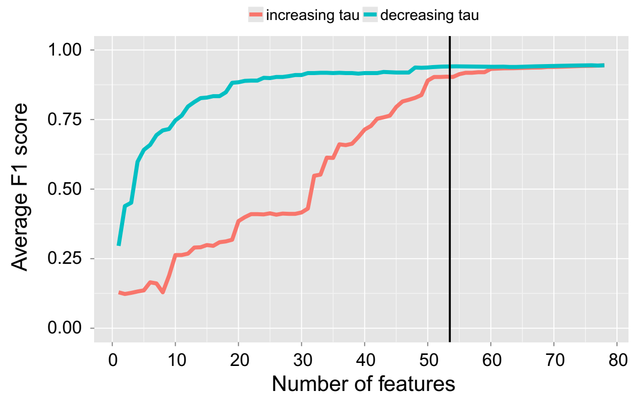


Fig. 5 Average F score for initial classifier for fivefold cross-validation for various number of features. *Blue line* shows the change in F score, while adding features from the most to the least useful one, and the *red line* shows the increase with the reversed order. The *vertical line* marks the feature set chosen for the final classifier (color figure online)

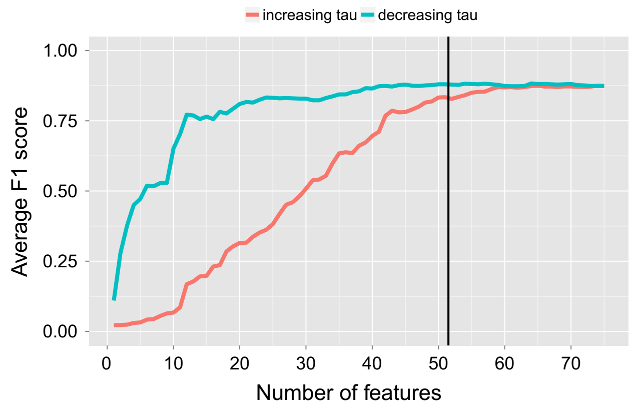


Fig. 6 Average F score for metadata classifier for fivefold cross-validation for various number of features. *Blue line* shows the increase in F score while adding features from the most to the least useful one, and the *red line* shows the increase with the reversed order. The *vertical line* marks the feature set chosen for the final classifier (color figure online)

4.2.2 SVM parameters adjustment

SVM parameters were also estimated using the zone validation dataset. The feature vectors were scaled linearly to interval [0, 1] according to the bounds found in the learning samples. In order to find the best parameters for the classifiers we performed a grid search over a three-dimensional space $\langle K, \Gamma, C \rangle$, where K is a set of kernel function types (linear, fourth degree polynomial, radial-basis and sigmoid), $\Gamma = \{2^i | i \in [-15, 3]\}$ is a set of possible values of the kernel coefficient γ, and $C = \{2^i | i \in [-5, 15]\}$ is a set of possible values of the penalty parameter. For every combination of the parameters, we performed a fivefold cross-validation. Finally, we chose those parameters, for which we obtained the highest mean F score (calculated as an average for individual classes). We also used classes weights based on the number of their training samples to set larger penalty for less represented classes.

Parameters for the best obtained results are presented in Tables 3 and 4. In both cases, we chose radial-basis kernel function, and chosen values of C and γ parameters are 2^5 and 2^{-3} in the case of initial classifier and 2^9 and 2^{-3} in the case of metadata classifier.

4.3 Metadata extraction

The purpose of this phase is to analyse zones labelled as *metadata* and extract a rich set of document's metadata information, including: title, authors, affiliations, relations author–affiliation, email addresses, relations author–email, abstract, keywords, journal, volume, issue, pages range, year and DOI.

The phase contains two steps:

1. Metadata zone classification (B1)—assigning specific metadata classes to metadata zones, described in detail in Sect. 4.2.
2. Metadata extraction (B2)—extracting atomic information from labelled zones.

During the last step (B2), a set of simple heuristic-based rules is used to perform the following operations:

– zones labelled as *abstract* are concatenated,
– as type is often specified just above the title, it is removed from the *title* zone if needed (based on a dictionary of types),
– authors, affiliations and keywords lists are split with the use of a list of separators,
– affiliations are associated with authors based on indexes and distances,
– email addresses are extracted from *correspondence* and *affiliation* zones using regular expressions,
– email addresses are associated with authors based on author names,
– pages ranges placed directly in *bib_info* zones are parsed using regular expressions,
– if there is no pages range given explicitly in the document, we also try to retrieve it from the pages numbers on each page,
– dates are parsed using regular expressions,
– journal, volume, issue and DOI are extracted from *bib_info* zones based on regular expressions.

Table 3 The results of SVM parameters searching for initial classification

Initial classification				
Kernel	Linear	4th poly.	RBF	Sigmoid
$\log_2(C), \log_2(\gamma)$	7, 1	9, −5	5, −3	15, −13
Mean F1 (%)	90.7	93.5	93.9	90.1

The table shows the mean F score values for all kernel function types obtained during fivefold cross-validation, as well as related values of C and γ parameters

Table 4 The results of SVM parameters searching for metadata classification

Metadata classification				
Kernel	Linear	4th poly.	RBF	Sigmoid
$\log_2(C), \log_2(\gamma)$	4, −9	7, −4	9, −3	11, −7
Mean F1 (%)	85.0	87.5	88.6	81.0

The table shows the mean F score values for all kernel function types obtained during fivefold cross-validation, as well as related values of C and γ parameters

4.4 Bibliography extraction

The goal of bibliography extraction is to extract a list of bibliographic references with their metadata (including *author*, *title*, *source*, *volume*, *issue*, *pages* and *year*) from zones labelled as *references*.

Bibliography extraction path contains two steps:

1. Reference strings extraction (C1)—dividing the content of references zones into individual reference strings.
2. Reference parsing (C2)—extracting metadata from reference strings.

4.4.1 Extracting reference strings

References zones contain a list of reference strings, each of which can span over one or more text lines. The goal of reference strings extraction is to split the content of those zones into individual reference strings. This step utilizes unsupervised machine learning techniques, which allows to omit time-consuming training set preparation and learning phases, while achieving very good extraction results.

Every bibliographic reference is displayed in the PDF document as a sequence of one or more text lines. Each text line in a reference zone belongs to exactly one reference string, some of them are first lines of their reference, others are inner or last ones. The sequence of all text lines belonging to bibliography section can be represented by the following regular expression:

```
(
 <first line of a reference>
 (
   <inner line of a reference>*
   <last line of a reference>
 )?
)*
```

In order to group text lines into consecutive references, first we determine which lines are first lines of their references. A set of such lines is presented in Fig. 7. To achieve this, we transform all lines to feature vectors and cluster them into two sets (first lines and all the rest). We make use of a simple observation that the first line from all references blocks is also the first line of its reference. Thus, the cluster containing this first line is assumed to contain all first lines. After recognizing all first lines, it is easy to concatenate lines to form consecutive reference strings.

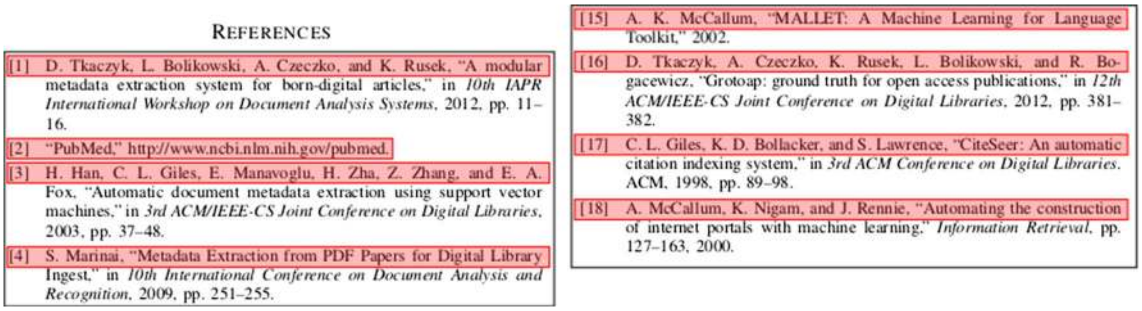
For clustering lines, we use KMeans algorithm with Euclidean distance metric. In this case $K = 2$, since the line set is clustered into two subsets. As initial centroids, we set the first line's feature vector and the vector with the largest distance to the first one. We use five features based on line relative length, line indentation, space between the line and the previous one, and the text content of the line (if the line starts with an enumeration pattern, if the previous line ends with a dot).

4.4.2 Reference strings parsing

Reference strings extracted from references zones contain important reference metadata. In this step, metadata is extracted from reference strings and the result is the list of document's parsed bibliographic references. The information we extract from the strings include: *author*, *title*, *source*, *volume*, *issue*, *pages* and *year*. An example of a parsed reference is shown in Fig. 8.

First a reference string is tokenized. The tokens are then transformed into vectors of features and classified by a supervised classifier. Finally, the neighbouring tokens with the same label are concatenated, the labels are mapped into final metadata classes and the resulting reference metadata record is formed.

The heart of the implementation is a classifier that assigns labels to reference tokens. For better performance, the classifier uses slightly more detailed labels than the target ones: *first_name* (author's first name or initial), *surname* (author's surname), *title*, *source* (journal or conference name), *volume*, *issue*, *page_first* (the lower bound of pages range), *page_last* (the upper bound of pages range), *year* and *text* (for separators and other tokens without a specific label). The token classifier employs conditional random fields and is built on top of GRMM and MALLET packages [35].


REFERENCES

[1] D. Tkaczyk, L. Bolikowski, A. Czeczko, and K. Rusek, "A modular metadata extraction system for born-digital articles," in *10th IAPR International Workshop on Document Analysis Systems*, 2012, pp. 11–16.

[2] "PubMed," http://www.ncbi.nlm.nih.gov/pubmed.

[3] H. Han, C. L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E. A. Fox, "Automatic document metadata extraction using support vector machines," in *3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, 2003, pp. 37–48.

[4] S. Marinai, "Metadata Extraction from PDF Papers for Digital Library Ingest," in *10th International Conference on Document Analysis and Recognition*, 2009, pp. 251–255.

[15] A. K. McCallum, "MALLET: A Machine Learning for Language Toolkit," 2002.

[16] D. Tkaczyk, A. Czeczko, K. Rusek, L. Bolikowski, and R. Bogacewicz, "Grotoap: ground truth for open access publications," in *12th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2012, pp. 381–382.

[17] C. L. Giles, K. D. Bollacker, and S. Lawrence, "CiteSeer: An automatic citation indexing system," in *3rd ACM Conference on Digital Libraries*. ACM, 1998, pp. 89–98.

[18] A. McCallum, K. Nigam, and J. Rennie, "Automating the construction of internet portals with machine learning," *Information Retrieval*, pp. 127–163, 2000.

Fig. 7 A fragment of the references section of an article. *Marked lines* are the first lines of their references. After detecting these lines, the references section content can be easily split to form consecutive references strings

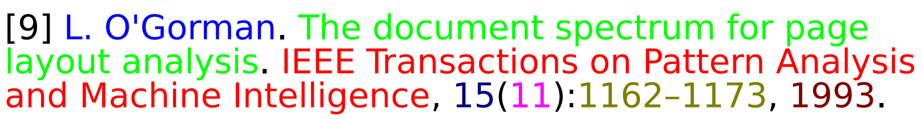

[9] L. O'Gorman. The document spectrum for page layout analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(11):1162–1173, 1993.

Fig. 8 An example of a bibliographic reference with various metadata information highlighted using *different colors*, and these are in order: *author*, *title*, *journal*, *volume*, *issue*, *pages* and *year* (color figure online)

CRF classifiers are a state-of-the-art technique for citation parsing. They achieve very good results for classifying instances that form a sequence, especially when the label of one instance depends on the labels of previous instances.

The basic features are the tokens themselves. We use 42 additional features to describe the tokens:

- Some of them are based on the presence of a particular character class, e.g. digits or lowercase/uppercase letters.
- Others check whether the token is a particular character (e.g. a dot, a square bracket, a comma or a dash), or a particular word.
- Finally, we use features checking if the token is contained by the dictionary built from the dataset, e.g. a dictionary of cities or words commonly appearing in the journal title.

It is worth to notice that the token's label depends not only on its feature vector, but also on the features of the surrounding tokens. To reflect this in the classifier, the token's feature vector contains not only features of the token itself, but also features of two preceding and two following tokens.

After token classification, fragments labelled as *first_name* and *surname* are joined together based on their order to form consecutive author names, and similarly fragments labelled as *page_first* and *page_last* are joined together to form pages range. Additionally, in the case of *title* or *source* labels, the neighbouring tokens with the same label are concatenated.

The result of bibliography extraction is a list of document's bibliographic references in a structured form, each of which contains the raw text as well as additional metadata.

5 Evaluation

We performed the evaluation of the key steps of the algorithm and the entire extraction process as well. The ground truth data used for the evaluation is based mainly on the resources of PubMed Central Open Access Subset [1].

Evaluated steps include: page segmentation (Sect. 5.2), initial and metadata zone classification (Sect. 5.3) and reference parsing (Sect. 5.4). Other steps were not directly evaluated, mainly due to the fact that creating ground truth datasets for them would be difficult and time-consuming. Since all the steps affect the final extraction result, they were all evaluated indirectly by the assessment of the performance of the entire CERMINE system (Sect. 5.5) and the comparison with similar tools as well (Sect. 5.6).

5.1 Datasets preparation

Table 5 provides details about all the datasets used for the experiments. In general, we use three types of data. Subsets of PubMed Central were used directly to evaluate the entire extraction workflow (metadata test set) and compare the performance of CERMINE with similar systems (comparison test set). Additionally, PMC resources served as a base for constructing GROTOAP and GROTOAP2 datasets, which were used for the experiments related to page segmentation (segmentation test set) and zone classification (zone validation set and zone test set). A set used for citation parser evaluation was build using PMC and also CiteSeer [36] and Cora-ref [37] (citation test set).

PubMed Central Open Access Subset [1] contains life sciences publications in PDF format, and their corresponding metadata in the form of NLM JATS files. NLM files contain a rich set of document's metadata (title, authors, affiliations, abstract, journal name, etc.), full text (sections, section titles, paragraphs, tables, equations) and also document's bibliography. Subsets of PMC were used to: (1) evaluate the entire metadata and references extraction workflow (metadata test set) and (2) compare the system performance with other tools (comparison test set).

Unfortunately, the quality of data in ground truth NLM JATS files varies from perfectly labelled documents to documents containing no valuable information at all. In some cases, NLM files lack the entire sections of the document (usually the bibliography and/or the body). Such files were filtered out in both sets, and for evaluation we used only documents, whose metadata files contained all three important sections: front matter, body and bibliography.

What is more, ground truth files from PMC contain only the annotated text of the document and do not preserve geometric features related to the way the text is displayed in PDF files. As a result, PMC could not be directly used for training and evaluation of the individual steps, such as page segmentation and zone classification. For these tasks, we built GROTOAP [38] and GROTOAP2 [4] datasets.

GROTOAP is a dataset of 113 documents in TrueViz format preserving not only the text content, but also geometric features of the text and zone labels. GROTOAP was built semi-automatically from PMC resources. First PDF documents were processed by automatic tools in order to extract the geometric structure along with zone labels, and the results were corrected manually by human experts. Since the task of correcting the geometric structure and zone labelling of the entire document is time-consuming, we were able to produce only a small set of documents. GROTOAP was used to evaluate page segmentation (segmentation test set).

GROTOAP2 is a successor of GROTOAP. GROTOAP2 is a much larger and diverse dataset, also containing information related to the document's text, geometric features and zone labels. The label set in GROTOAP2 is a union of all labels used in both zone classifiers.

GROTOAP2 was created semi-automatically using PMC resources (Fig. 9). Our goal was to create a fairly large dataset, useful for machine learning algorithms. Unfortu-

Table 5 The summary of all the datasets used in the experiments

Name	Source	Format	Content	Purpose
Segmentation test set	GROTOAP	TrueViz	113 documents	The evaluation of page segmentation (Sect. 5.2)
Zone validation set	GROTOAP2	TrueViz	100 documents containing 14,000 labelled zones, 2743 of which are metadata zones	Zone classifiers feature selection (Sect. 4.2.1) and SVM parameters determination (Sect. 4.2.2)
Zone test set	GROTOAP2	TrueViz	2551 documents containing 355,779 zones, 68,557 of which are metadata zones	Zone classifiers evaluation (Sect. 5.3) and final classifiers training
Citation test set	CiteSeer, Cora-ref and PMC	NLM JATS	4000 parsed citations (2000 from CiteSeer and Cora-ref, 2000 from 1991 different PMC documents)	The evaluation of the references parser (Sect. 5.4)
Metadata test set	PubMed Central	PDF + NLM JATS	47,983 PDF documents with corresponding metadata records	The evaluation of the entire metadata and bibliography extraction workflow (Sect. 5.5)
Comparison test set	PubMed Central	PDF + NLM JATS	1943 PDF documents with corresponding metadata records	The comparison of CERMINE's performance with the performance of other similar tools (Sect. 5.6)

nately, an approach used for GROTOAP would not allow to create a large dataset, due to the manual correction of every document. Instead, we decided to make use of the text labelling already present in the PMC's NLM JATS files to assign labels to zones automatically, while the zones themselves were constructed using CERMINE tools. More precisely, GROTOAP2 was created with the following steps:

1. First, PDF files from PMC were processed automatically by CERMINE in order to extract the hierarchical geometric structure and the reading order.
2. The text content of every zone was then compared to labelled text from NLM files with the use of Smith–Waterman sequence alignment algorithm [39]. This allowed to assign labels to zones.
3. Files with a lot of zones labelled as "unknown", that is zones, for which the labelling process was unable to find a concrete label, were filtered out.
4. A small sample of the remaining files was inspected manually. This resulted in identifying a number of repeated problems and errors in the dataset.
5. Based on the results of the analysis, we developed a set of heuristic-based rules and applied them to the dataset in order to increase the labelling accuracy.

More details about GROTOAP2 dataset and its creation process can be found in [4].

Since GROTOAP's creation process did not contain manual correction of every document, the dataset contains errors, caused by both segmentation and labelling steps. Segmentation errors were comparatively rare. According to the evaluation we performed on a random sample of 50 documents, the accuracy of zone labelling is 93 %. Despite this drawback, the lack of manual correction of every document guaranteed the scalability of the method, which allowed to create much larger dataset than in the case of more traditional approaches.

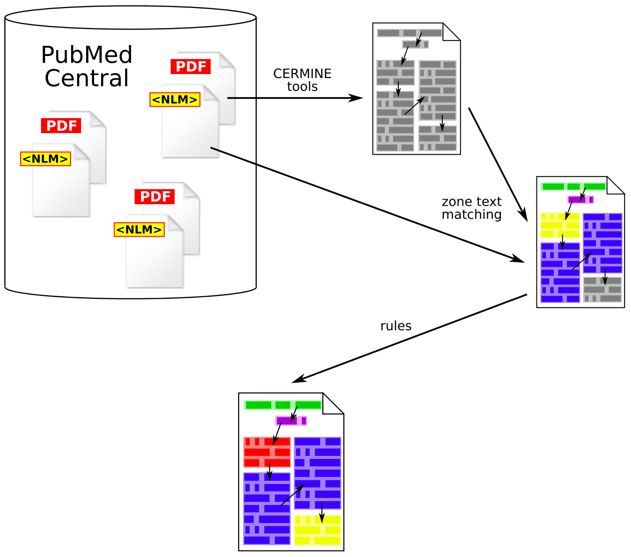


Fig. 9 Semi-automatic method of creating GROTOAP2 dataset. First automatic tools extracted the hierarchical geometric structure and the reading order of a document. Next, we automatically assigned labels to zones by matching their text to labelled fragments from NLM files. Finally, additional rules were developed manually and applied to the dataset in order to increase the labelling accuracy. It should be noted that since CERMINE was not involved in the process of assigning labels, the dataset can be used to evaluate the performance of zone classification

Since CERMINE was not involved in the process of assigning labels, subsets of GROTOAP2 could be used for the experiments with zone classification: feature selection and SVM parameters adjustment (zone validation set), and final zone classifiers evaluation and training (zone test set).

For reference parser evaluation, we used CiteSeer [36], Cora-ref [37] and PubMed Central resources combined together into a single set (citation test set).

CiteSeer and Cora-ref already contain parsed references. Unfortunately, due to some differences in the labels used, labels mapping had to be performed. Labels from original datasets were mapped in the following way: *title* and *year* remained the same; *journal*, *booktitle*, *tech* and *type* were mapped to *source*; *date* was mapped to *year*. Labels *author* and *pages* were split, respectively, into *givenname* and *surname*, *page_first* and *page_last* using regular expressions. All remaining tokens were labelled as *text*.

NLM files from PMC also contain parsed references. Unfortunately, in most cases, they do not preserve the entire reference strings from the original PDF file, and separators and punctuation are often omitted. For this reason, the reference set was built using a similar technique as in the case of GROTOAP2. We extracted reference strings from PDF files using CERMINE tools and labelled them using annotated data from NLM files.

5.2 Page segmentation

Page segmenter was evaluated using the entire GROTOAP dataset. For each structure type (zone, line, word), we calculated the overall accuracy over all documents that is the percentage of elements correctly constructed by the algorithm. An item is considered constructed correctly if it contains exactly the same set of characters as the original element. Since in our ground truth dataset every table and figure is placed in one zone, and Docstrum usually divides these (often sparse) areas into more zones, these regions were excluded from the evaluation.

We performed the evaluation of two versions of the segmentation algorithm: the original Docstrum and the algorithm with the modifications listed in Sect. 4.1.2. The results are shown in Fig. 10. For all structure types, the modifications resulted in increased extraction accuracy.

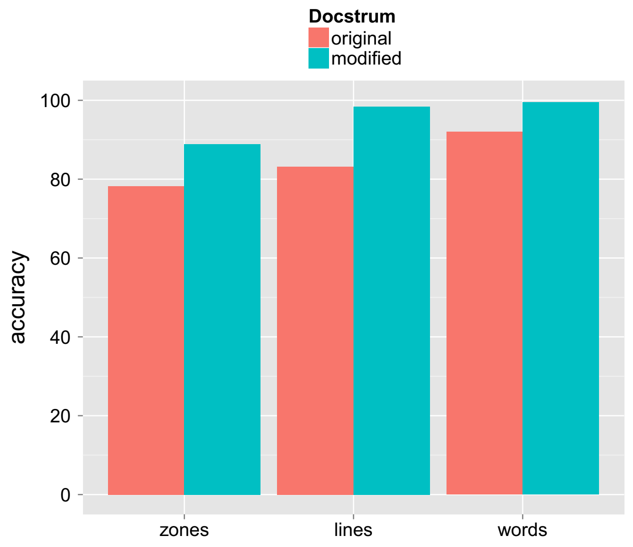


Fig. 10 The results of page segmentation evaluation. The *plot* shows the accuracy of extracting zones, lines and words for the original Docstrum algorithm and Docstrum with modifications proposed in Sect. 4.1.2

5.3 Zone classification

Both zone classifiers were evaluated by a fivefold cross-validation using zone test set (described in Sect. 5.1). The Tables 6 and 7 show the confusion matrices as well as precision and recall values for individual classes for initial and metadata classification, respectively.

For a class C, precision and recall were calculated in the following way:

$$\text{Precision}_C = \frac{|S_C|}{|C_C|}, \quad \text{Recall}_C = \frac{|S_C|}{|G_C|}$$

where S_C is a set of zones correctly recognized as C by the classifier, C_C is a set of zones labelled as C by the classifier and G_C is a set of zones labelled as C in the ground truth data.

Initial classifier achieved the following results calculated as mean values for individual classes: precision 97.2 %, recall 95.4 %, F score 96.3 %. The results achieved by metadata classifier were as follows: precision 95.4 %, recall 95.1 %, F score 95.3 %.

We also compared the performance of the classification obtained from our two classifiers executed in sequence with one combined classifier, which assigns both general categories and specific metadata classes (more details about the two approaches and the decision to use two classification steps instead of one can be found in Sect. 4.2). The combined classifier achieved 95.1 % accuracy and 85.2 % mean F score, while two separate classifiers working together (the current solution) achieved 95.3 % accuracy and 85.9 % F score. The performance of these approaches is thus very similar to each other.

5.4 Reference parsing

Bibliographic reference parser was evaluated with the use of a fivefold cross-validation on the citation test set (described in Sect. 5.1). For every metadata class, we computed precision and recall in a similar way as in the case of zone classification. This time the objects in S_C, C_C and G_C sets were not individual tokens, but entire reference substrings. As a consequence, a token correctly labelled with a class C contributes to the overall success rate only if the entire token sequence of class C containing the given token is correctly labelled.

Table 6 Confusion matrix for initial classification for fivefold cross-validation

	Metadata	Body	References	Other	Precision (%)	Recall (%)
Metadata	**66,042**	2181	75	259	96.6	96.3
Body	1551	**232,464**	177	934	97.9	98.9
References	47	806	**17,489**	67	98.2	95.0
Other	733	2118	65	**30,771**	96.1	91.3

Rows and columns represent the desired and obtained classification result, respectively
Bold values on the main matrix diagonal are the numbers of correctly classified zones of respective classes

Figure 11 shows precision and recall values for individual metadata classes. The parser achieved the following scores calculated as mean values for individual classes: precision 92.9 %, recall 93.8 %, F score 93.3 %.

5.5 Metadata extraction evaluation

The evaluation of the entire workflow was performed with the use of metadata test set (described in Sect. 5.1). The PDF files were processed by CERMINE and the resulting metadata (the "tested" documents) was compared to metadata stored in NLM files (the "ground truth" documents).

For each type of metadata, we used different measures of correctness. In general, we deal with two types of metadata fields: those that appear at most once per document (these are: title, abstract, journal, volume, issue, pages range, year and DOI) and those present as lists (authors, affiliations, email addresses, keywords and bibliographic references).

In the first case, for every document, a single string from NLM file was compared to the extracted string, which gives a binary output: information extracted correctly or not. The overall precision and recall scores for a metadata class C are calculated in the following way:

$$\text{Precision}_C = \frac{|S_C|}{|C_C|}, \quad \text{Recall}_C = \frac{|S_C|}{|G_C|}$$

where S_C is a set of documents from which the non-empty information of a class C was correctly extracted, C_C is a set of tested documents with non-empty field of class C, and finally G_C is a set of ground truth documents with non-empty field of class C.

Some information types from this group, such as article's volume, issue, DOI, dates and pages, were considered correct only if exactly equal to NLM data. As the journal name is often abbreviated, we marked it as correct if it was a subsequence of the ground truth journal name. Article's title and abstract were tokenized and compared with the use of Smith–Waterman sequence alignment algorithm [39].

In the case of list metadata types, for every document the elements of tested and ground truth lists were compared using cosine distance. This resulted in individual precision and recall for every document. The overall precision and recall were computed as mean values over all documents. In the case of bibliographic references, only their full text was compared, and the detailed metadata was ignored.

The evaluation results are shown in Fig. 12. CERMINE achieved the following results calculated as mean values for individual metadata classes: precision 81.0 %, recall 74.7 %, F score 77.5 %.

5.6 Comparison evaluation

Comparison test set (described in Sect. 5.1) was used to compare the performance of CERMINE with similar extraction systems. The results are shown in Table 8. The evaluation methodology was the same as before, with the exception of ParsCit system. Since ParsCit analyses only the text content of a document, PDF files were first transformed to text using `pdftotext` tool. What is more, the output of ParsCit can contain multiple titles or abstracts; thus, for this system, all metadata classes were treated as list types.

For most metadata classes, CERMINE performs the best. The worst values were obtained in the case of ParsCit system, which was probably caused by the fact that the algorithm inspects only the text content of a documents, ignoring hints related to the way the text is displayed in the PDF file.

5.7 Error analysis

The errors made by the extraction workflow can be divided into two groups: metadata was not extracted or the extracted information is incorrect. The majority of errors happen in the following situations:

- When two (or more) zones with different roles in the document are placed close to each other, they are often merged together by the segmenter. In this case, the classification is more difficult and by design only one label is assigned to such a hybrid zone. A potential solution would be to introduce additional labels for pairs of labels that often appear close to each other, for example *title_author* or *author_affiliation*, and split the content of such zones later in the workflow.
- The segmenter introduces other errors as well, such as incorrectly attaching an upper index to the line above the current line, or merging text written in two columns. These

Table 7 Confusion matrix for metadata classification for fivefold cross-validation

	Abstract	Affiliation	Author	Bib_info	Correspondence	Dates	Editor	Keywords	Title	Type	Copyright	Precision (%)	Recall (%)
Abstract	**6866**	8	7	62	8	1	1	23	7	5	10	97.7	98.1
Affiliation	11	**3532**	16	31	62	5	8	3	1	6	6	95.5	96.0
Author	4	14	**2684**	42	18	0	3	1	6	6	4	96.9	96.5
Bib_info	75	22	14	**40,982**	25	119	1	41	16	115	100	98.7	98.9
Corresp.	9	107	15	32	**1616**	2	0	3	1	1	3	92.8	90.3
Dates	5	1	4	136	3	**2835**	0	1	0	2	13	94.7	94.5
Editor	0	2	1	0	0	0	**473**	0	0	0	0	96.9	99.4
Keywords	28	8	5	86	1	6	1	**896**	5	7	1	91.5	85.8
Title	9	0	13	26	0	0	0	3	**2574**	6	2	98.3	97.8
Type	4	0	4	88	0	2	1	6	6	**1497**	2	91.0	93.0
Copyright	14	5	7	45	8	23	0	2	3	0	**2927**	95.4	96.5

Rows and columns represent the desired and obtained classification result, respectively
Bold values on the main matrix diagonal are the numbers of correctly classified zones of respective classes

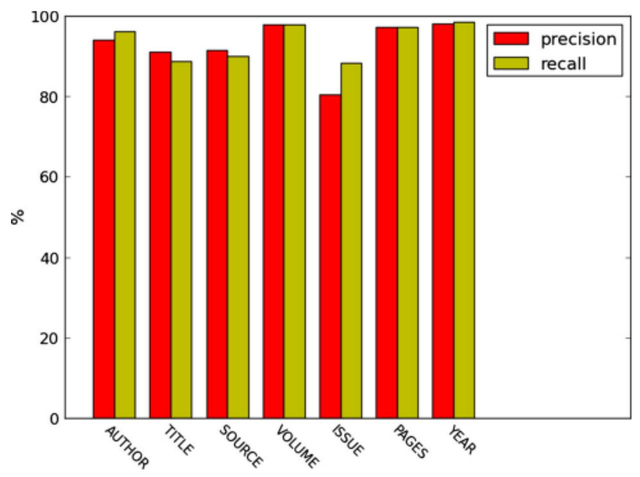


Fig. 11 Bibliographic reference parser evaluation. The figure shows precision and recall values for extracting reference fragments belonging to individual metadata classes. A given fragment is considered correctly extracted, if it is identical to the ground truth data

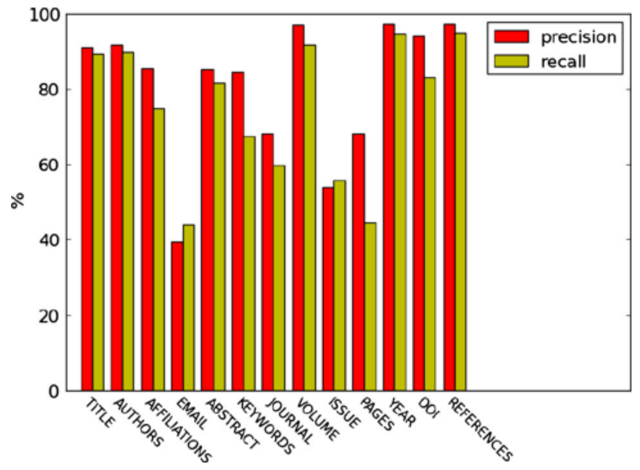


Fig. 12 The evaluation results of CERMINE's extraction process on metadata test set. The figure shows precision and recall values for individual metadata classes

errors can be corrected by further improvement of the page segmenter.
- Zone classification errors are also responsible for a lot of extraction errors. These errors can be improved by adding training instances to the training set and improving the labelling accuracy in GROTOAP2.
- Sometimes the metadata, usually keywords, volume, issue or pages, is not explicitly given in the input PDF file. Since CERMINE analyses the PDF file only, such information cannot be extracted. This is in fact not an extraction error. Unfortunately, since ground truth NLM data in PMC usually contains such information, whether it is written in the PDF or not, these situations also contribute to the overall error rates (equally for all evaluated systems).

Table 8 The results of comparing the performance of various metadata extraction systems

	CERMINE	PDFX	GROBID	ParsCit	Pdf-extract
Title	**95.5**	85.7	82.5	34.1	49.4
	93.4	84.7	77.4	39.6	49.4
	94.5	85.2	79.8	36.6	49.4
Authors	**90.2**	71.2	85.9	57.9	–
	89.0	71.5	**90.5**	48.6	–
	89.6	71.3	88.1	52.8	–
Affiliations	88.2	–	**90.8**	72.2	–
	83.1	–	51.8	44.3	–
	85.6	–	66.0	54.9	–
Email addresses	51.7	**53.0**	26.9	28.8	–
	42.6	**73.6**	7.8	36.2	–
	46.7	**61.6**	12.1	32.1	–
Abstract	**82.8**	71.1	70.4	47.7	–
	79.9	66.7	67.7	61.3	–
	81.3	68.8	69.0	53.7	–
Keywords	89.9	–	**94.2**	15.6	–
	63.5	–	44.2	3.0	–
	74.4	–	60.2	5.1	–
Journal	**80.3**	–	–	–	–
	73.2	–	–	–	–
	76.6	–	–	–	–
Volume	**93.3**	–	–	–	–
	83.0	–	–	–	–
	87.8	–	–	–	–
Issue	**53.7**	–	–	–	–
	28.4	–	–	–	–
	37.1	–	–	–	–
Pages	**87.0**	–	–	–	–
	80.4	–	–	–	–
	83.5	–	–	–	–
Year	**96.3**	–	95.7	–	–
	95.0	–	40.4	–	–
	95.6	–	56.8	–	–
DOI	98.2	–	**99.1**	–	–
	75.0	–	65.4	–	–
	85.1	–	78.8	–	–
References	**96.1**	91.3	79.7	81.2	80.4
	89.8	88.9	66.7	71.8	57.5
	92.8	90.1	72.6	76.2	67.0

In every cell, the precision, recall and *F* score values are shown. The best results in every category are bolded

The most common extraction errors include:

- Title merged with other parts of the document, when title zone is placed close to another region.
- Title not recognized, for example when it appears on the second page of the PDF file.
- Title zone split by the segmenter into a few zones, and only a subset of them is correctly classified.
- Authors zone not labelled, in that case the authors are missing.
- Authors zone merged with other fragments, such as affiliations or research group name, in such cases additional fragments appear in the authors list.

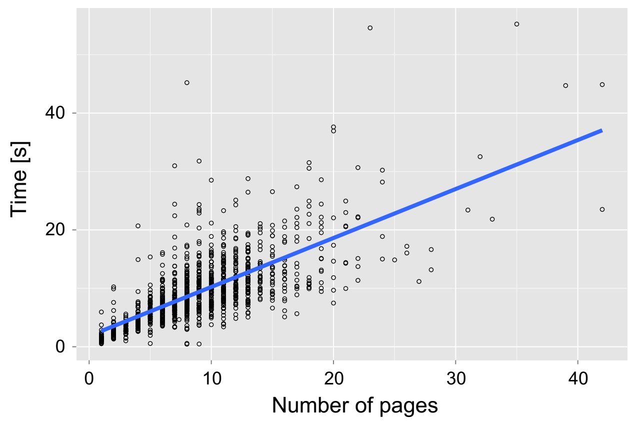


Fig. 13 The plot shows CERMINE's processing time (in seconds) as a function of the number of pages of a document for a subset of 1238 documents from PMC

- Affiliation zone not properly recognized by the classifier, for example when it not visually separated from other zones, or placed at the end of the document. Affiliations are missing in that case.
- The entire abstract or a part of it recognized as *body* by the classifier, as a result the abstract or a part of it is missing.
- The first *body* paragraph recognized incorrectly as *abstract*, as a result the extracted abstract contains a fragment of the document's proper text.
- Bibliographic information missing from a PDF file or not recognized by the classifiers, as a result journal name, volume, issue and/or pages range are not extracted.
- Keywords missing because the zone was not recognized or not included in the PDF file.
- A few of the references zones classified as *body*, and in such cases some or all of the references are missing.

5.8 Processing time

The processing time of a document depends mainly on its number of pages. The most time-consuming steps are page segmentation and initial zone classification.

Figure 13 shows the processing time as a function of the number of document's pages for 1238 random documents. The average processing time for this subset was 9.4 s.

6 Conclusions and future work

The article presents CERMINE—a system for extracting both metadata and bibliography from scientific articles in a born-digital form. CERMINE is very useful for digital libraries and similar environments whenever they have to deal with documents with metadata information missing, fragmentary or not reliable. Automatic extraction tools provided by CERMINE support a number of tasks such as intelligent searching, finding similar and related documents, building citation and author networks, and so on.

The system is open source and available online at http://cermine.ceon.pl. The modular architecture and the use of supervised and unsupervised machine learning techniques make CERMINE flexible and easy to adapt to new document layouts. The evaluation against a large and diverse dataset shows good results for the key individual steps and the entire extraction workflow. For most metadata types, the results are better than in the case of other similar extraction systems.

Our future plans include:

- extending the workflow, so that the system is able to process documents in the form of scanned pages as well,
- expanding the workflow architecture by adding a process path for extracting structured full text containing sections and subsections, headers and paragraphs,
- adding affiliation parsing step, the goal of which is to extract affiliation metadata: institution name, address and country,
- making the citation dataset used for parser evaluation publicly available.

Acknowledgments This work has been supported by the European Commission as part of the FP7 OpenAIREplus (Grant No. 283595) and OCEAN projects.

References

1. PubMed. http://www.ncbi.nlm.nih.gov/pubmed
2. CERMINE. http://cermine.ceon.pl
3. Tkaczyk, D., Szostek, P., Dendek, P.J., Fedoryszak, M., Bolikowski, L.: CERMINE—automatic extraction of metadata and references from scientific literature. In: 11th IAPR International Workshop on Document Analysis Systems, pp. 217–221 (2014)
4. Tkaczyk, D., Szostek, P., Bolikowski, L.: GROTOAP2—the methodology of creating a large ground truth dataset of scientific articles. D-Lib Magazine (2014)
5. Giuffrida, G., Shek, E.C., Yang, J.: Knowledge-based metadata extraction from postscript files. In: ACM DL, pp. 77–84 (2000)
6. Constantin, A., Pettifer, S., Voronkov, A.: PDFX: fully-automated pdf-to-xml conversion of scientific literature. In: ACM Symposium on Document Engineering, pp. 177–180 (2013)
7. Pdf-extract. http://labs.crossref.org/pdfextract/
8. Han, H., Giles, C.L., Manavoglu, E., Zha, H., Zhang, Z., Fox, E.A.: Automatic document metadata extraction using support vector machines. In: ACM/IEEE 2003 Joint Conference on Digital Libraries, pp. 37–48 (2003)

9. Kovacevic, A., Ivanovic, D., Milosavljevic, B., Konjovic, Z., Surla, D.: Automatic extraction of metadata from scientific publications for CRIS systems. Program **45**(4), 376–396 (2011)
10. Lu, X., Kahle, B., Wang, J.Z., Giles, C.L.: A metadata generation system for scanned scientific volumes. In: ACM/IEEE Joint Conference on Digital Libraries, JCDL 2008, Pittsburgh, PA, USA, 16–20 June 2008, pp. 167–176 (2008)
11. Marinai, S.: Metadata extraction from PDF papers for digital library ingest. In: 10th International Conference on Document Analysis and Recognition, pp. 251–255 (2009)
12. Cui, B., Chen, X.: An improved hidden Markov model for literature metadata extraction. In: Advanced Intelligent Computing Theories and Applications, 6th International Conference on Intelligent Computing, pp. 205–212 (2010)
13. Kern, R., Jack, K., Hristakeva, M., Granitzer, M.: Teambeam—meta-data extraction from scientific literature. D Lib Mag. **18**(7/8), 1 (2012)
14. Lopez, P.: GROBID: combining automatic bibliographic data recognition and term extraction for scholarship publications. In: Research and Advanced Technology for Digital Libraries, 13th European Conference, pp. 473–474 (2009)
15. Luong, M., Nguyen, T.D., Kan, M.: Logical structure recovery in scholarly articles with rich document features. IJDLS **1**(4), 1–23 (2010)
16. Councill, I.G., Giles, C.L., Kan, M.: Parscit: an open-source CRF reference string parsing package. In: Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May–1 June 2008, Marrakech, Morocco (2008)
17. Gao, L., Tang, Z., Lin, X.: CEBBIP: a parser of bibliographic information in chinese electronic books. In: Proceedings of the 2009 Joint International Conference on Digital Libraries, pp. 73–76 (2009)
18. Zou, J., Le, D.X., Thoma, G.R.: Locating and parsing bibliographic references in HTML medical articles. IJDAR **13**(2), 107–119 (2010)
19. Day, M., Tsai, R.T., Sung, C., Hsieh, C., Lee, C., Wu, S., Wu, K., Ong, C., Hsu, W.: Reference metadata extraction using a hierarchical knowledge representation framework. Decis. Support Syst. **43**(1), 152–167 (2007)
20. Vilarinho, E.C.C., da Silva, A.S., Gonçalves, M.A., de Sá Mesquita, F., de Moura, E.S.: FLUX-CIM: flexible unsupervised extraction of citation metadata. In: ACM/IEEE Joint Conference on Digital Libraries, pp. 215–224 (2007)
21. Zhang, Q., Cao, Y., Yu, H.: Parsing citations in biomedical articles using conditional random fields. Comput. Biol. Med. **41**(4), 190–194 (2011)
22. Zhang, X., Zou, J., Le, D.X., Thoma, G.R.: A structural SVM approach for reference parsing. BMC Bioinform. **12**(S–3), S7 (2011)
23. Hetzner, E.: A simple method for citation metadata extraction using hidden markov models. In: ACM/IEEE Joint Conference on Digital Libraries, pp. 280–284 (2008)
24. PDFX. http://pdfx.cs.man.ac.uk/
25. Grobid. https://github.com/grobid/grobid
26. ParsCit. http://aye.comp.nus.edu.sg/parsCit/
27. Lee, C.H., Kanungo, T.: The architecture of trueviz: a groundtruth/metadata editing and visualizing toolkit. Pattern Recognit. **36**(3), 811–825 (2003)
28. NLM, JATS. http://dtd.nlm.nih.gov/archiving/
29. NISO Z39.96-2012. http://www.niso.org/apps/group_public/document.php?document_id=10591
30. iText. http://itextpdf.com/
31. O'Gorman, L.: The document spectrum for page layout analysis. IEEE Trans. Pattern Anal. Mach. Intell. **15**(11), 1162–1173 (1993)
32. PdfMiner. http://www.unixuser.org/euske/python/pdfminer/
33. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. ACM TIST **2**(3), 27 (2011)
34. Goodman, L.A., Kruskal, W.H.: Measures of association for cross classifications iii: approximate sampling theory. J. Am. Stat. Assoc. **58**, 310–364 (1963)
35. McCallum, A.K.: MALLET: a machine learning for language toolkit (2002)
36. Giles, C.L., Bollacker, K.D., Lawrence, S.: Citeseer: an automatic citation indexing system. In: Proceedings of the 3rd ACM International Conference on Digital Libraries, pp. 89–98 (1998)
37. McCallum, A., Nigam, K., Rennie, J.: Automating the construction of internet portals with machine learning. Inf. Retr. **3**, 127–163 (2000)
38. Tkaczyk, D., Czeczko, A., Rusek, K., Bolikowski, L., Bogacewicz, R.: GROTOAP: ground truth for open access publications. In: Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 381–382 (2012)
39. Smith, T., Waterman, M.: Identification of common molecular subsequences. J. Mol. Biol. **147**(1), 195–197 (1981)