Extracting Contextual Information from Scientific Literature Using CERMINE System

Dominika Tkaczyk(✉) and Łukasz Bolikowski

Interdisciplinary Centre for Mathematical and Computational Modelling,
University of Warsaw, Warsaw, Poland
{d.tkaczyk,l.bolikowski}@icm.edu.pl

Abstract. CERMINE is a comprehensive open source system for extracting structured metadata and references from born-digital scientific literature. Among other information, the system is able to extract information related to the context the article was written in, such as the authors and their affiliations, the relations between them or references to other articles. Extracted information is presented in a structured, machine-readable form. CERMINE is based on a modular workflow, whose loosely coupled architecture allows for individual components evaluation and adjustment, enables effortless improvements and replacements of independent parts of the algorithm and facilitates future architecture expanding. The implementation of the workflow is based mostly on supervised and unsupervised machine-learning techniques, which simplifies the procedure of adapting the system to new document layouts and styles. In this paper we outline the overall workflow architecture, describe key aspects of the system implementation, provide details about training and adjusting of individual algorithms, and finally report how CERMINE was used for extracting contextual information from scientific articles in PDF format in the context of ESWC 2015 Semantic Publishing Challenge. CERMINE system is available under an open-source licence and can be accessed at http://cermine.ceon.pl.

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. 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.

Semantic publishing addresses these issues by the enhancement of scholarly data with metadata and interlinking, allowing the machines to better understand the structure, meaning and relations of published information. Machine-readable

F. Gandon et al. (Eds.): SemWebEval 2015, CCIS 548, pp. 93–104, 2015.
DOI: 10.1007/978-3-319-25518-7_8

metadata describing scholarly communication, for example metadata related to citations, authors, organizations, research centres, projects or datasets, facilitates solving tasks like building citation and author networks, providing useful tools for intelligent search, detecting similar and related documents and authors, the assessment of the achievements of individual authors and entire organizations, identifying people and teams with a given research profile, and many more.

Unfortunately, in practice good quality metadata is not always available, sometimes it is missing, full of errors or fragmentary. In such cases there is a need to automatically extract the metadata directly from source documents, often stored in PDF format. Such automatic analysis of PDF documents is 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 [5] contains publications from nearly 500 different publishers, many of which use original layouts and styles in their articles. What is more, PDF format does not preserve the information related to the document's structure, such as words and paragraphs, lists and enumerations, 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 [13] — a comprehensive, open-source tool for automatic metadata extraction from born-digital scientific literature. CERMINE's extraction algorithm performs a thorough analysis of the input scientific publication in PDF format and extracts:

- a rich set of the document's metadata, including the title, authors, their affiliations, emails, abstract, keywords, year of publication, etc.,
- a list of bibliographic references along with their metadata.

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. This was achieved by employing supervised and unsupervised machine-learning algorithms trained on large and diverse datasets. It also resulted in increased maintainability of the system, as well as its ability to adapt to previously unseen document layouts.

CERMINE is based on a modular workflow composed of a number of steps with carefully defined input and output. By virtue of such workflow architecture individual steps can be maintained separately, making it easy to perform evaluation or training, improve or replace one step implementation without changing other parts of the workflow. CERMINE web service, as well as the source code, can be accessed online at http://cermine.ceon.pl [14].

The system is participating in ESWC 2015 Semantic Publishing Challenge. Its task is to mine PDF articles from CEUR workshop proceedings in order to extract the information related to the context in which the papers were written.

This article describes the overall extraction workflow architecture and key steps implementations, provides details about the training of machine-learning based algorithms, and finally reports how CERMINE was used for information extraction in the context of ESWC 2015 Semantic Publishing Challenge.

2 System Overview

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 in NLM JATS format [3]: the document's metadata and bibliography.

CERMINE's web service can be accessed at http://cermine.ceon.pl. The code is available on GitHub at https://github.com/CeON/CERMINE. The system provides also a REST service that allows for executing the extraction process by machines. It can be accessed using cURL tool:

```
$ curl -X POST --header "Content-Type: application/binary" -v \
  --data-binary @article.pdf http://cermine.ceon.pl/extract.do
```

2.1 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. 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.

The inner model of the document used during CERMINE's analysis 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 structure level and labels describing the role of the zones. The model can be serialized using XML TrueViz format.

The original output format of the extraction process is NLM JATS [3]. JATS (Journal Article Tag Suite) defines a rich set of XML elements and attributes for describing scientific publications. Documents in JATS format can store a wide range of structured metadata of the document, hierarchical full text and the bibliography in the form of a list of references along with their metadata.

Recently added functionality, essential for the semantic publishing challenge, is exporting information extracted from a set of articles as LOD dataset in RDF format. Currently exported dataset contains only the relevant subset of extracted metadata.

2.2 System Architecture

CERMINE's extraction workflow (Fig. 1) is composed of the following stages:

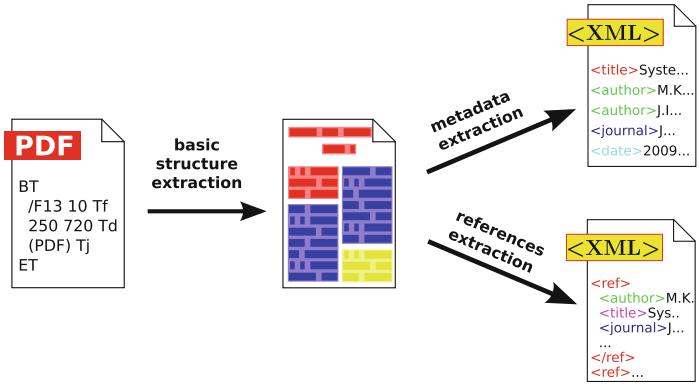

Fig. 1. CERMINE's extraction workflow architecture. At the beginning the geometric structure is extracted from the input PDF file. Then metadata and bibliography are extracted in two parallel paths.

[A] **Basic structure extraction** stage — analysing the input PDF file in order to construct its geometric hierarchical representation by executing the following steps:
  - [A1] **Character extraction** — extracting individual characters with their coordinates and dimensions from the input PDF using iText library [2].
  - [A2] **Page segmentation** — constructing the document's geometric hierarchical structure containing pages, zones, lines, words and characters, using enhanced Docstrum algorithm [11].
  - [A3] **Reading order resolving** — determining the reading order for all structure elements using bottom-up heuristic-based algorithm.
  - [A4] **Initial zone classification** — classifying the document's zones into categories: *metadata*, *body*, *references* and *other* using Support Vector Machines classifier.

[B] **Metadata extraction** stage — extracting a rich set of document's metadata from zones labelled as *metadata* by executing the following steps:
  - [B1] **Metadata zone classification** — classifying the document's zones into specific metadata classes using Support Vector Machines classifier.
  - [B2] **Metadata extraction** — extracting atomic metadata information from labelled zones using a list of simple rules.
  - [B3] **Affiliation parsing** — identifying organization, address and country in affiliation strings using Conditional Random Fields classifier.

[C] **Bibliography extraction** stage — extracting a list of parsed citations from zones labelled as *references* by executing the following steps:
  - [D1] **Reference extraction** — dividing the content of *references* zones into individual reference strings using K-Means clustering.
  - [D2] **Reference parsing** — extracting metadata information from references strings using Conditional Random Fields token classifier.

3 Metadata Extraction Algorithms

This section provides details about the implementations of key steps of the workflow. More information about the system implementation can be found in [13].

3.1 Geometric Structure Extraction

Structure extraction 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 also features related to the way the text is displayed in the PDF file.

Geometric structure extraction is composed of three steps:

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

The purpose of character extraction is to extract individual characters from the PDF stream along with their positions on the page, widths and heights. The implementation is based on open-source iText [2] 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 characters and their individual widths and positions are calculated.

The goal of page segmentation is to create a geometric hierarchical structure storing the document's content, consisting of pages, zones, lines, words and characters. Page segmentation is implemented with the use of a modified bottom-up Docstrum algorithm [11]. In this approach, the histograms of nearest-neighbor pairs of individual characters are analyzed in order to estimate the text orientation angle and also within-line and between-line spacings. This allows to determine text lines and finally group lines into zones.

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. The 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 using heuristics taken from [4], making use of an observation that the natural reading order descends from top to bottom, if successive zones are aligned vertically, otherwise it traverses from left to right.

3.2 Content Classification

The goal of content classification is to label each zone with a functional class. The classification is done in two stages: initial classification assigns general categories

(*metadata*, *references*, *body*, *other*), while the goal of metadata classification is to classify all metadata zones into specific metadata classes (*abstract*, *bib info*, *type*, *title*, *affiliation*, *author*, *correspondence*, *dates*, *editor*, *keywords*). Content classification is a crucial stage of the entire analysis and, along with page segmentation have the biggest impact on the extraction results.

Both classifiers use Support Vector Machines and their implementation is based on LibSVM library [7]. The classifiers differ in SVM parameters, but in both cases the best parameters were found by performing a grid-search using a set of 100 documents from PubMed Central Open Access Subset (PMC) and maximizing mean F1 score obtained during a 5-fold cross validation.

In order to perform zone classification, each zone is transformed into a vector of feature values, which are to a great extent the same for both classifiers. The initial and metadata classifiers use 83 and 62 features, respectively:

- geometrical — based on attributes such as the dimensions and coordinates, distance to the nearest zone, free space below and above the zone, etc.,
- lexical — based upon keywords characteristic for different parts of narration, that is: affiliation, acknowledgment, abstract, references, article type, etc.,
- sequential — based on sequence-related information, eg. class of the previous zone, presence of the same text blocks on the surrounding pages, etc.,
- formatting — eg. font size in the current and adjacent zones, the amount of blank space inside zones etc.,
- heuristics — eg. uppercase word count, percentage of numbers in a text block, if each line starts with enumeration-like tokens, etc.

3.3 Author and Affiliation Extraction

As a result of classifying the document's fragments, we usually obtain a few regions labelled as *author* or *affiliation*. In this step we extract individual author names and affiliations and determine relations between them.

In general the implementation is based on heuristics and regular expressions, but the details depend on the article's layout. There are two main styles used in different layouts: (1) author names are grouped together in a form of a list, and affiliations are also placed together below the author's list, at the bottom of the first page or even just before the bibliography section (an example is shown in Fig. 2), and (2) each author is placed in a separate zone along with its affiliation and email address (an example is shown in Fig. 3).

In the case of a layout of the first type (Fig. 2), at the beginning authors' lists are split using a predefined lists of separators. Then we detect affiliation indexes based on predefined lists of symbols and also geometric features. Detected indexes are then used to split affiliation lists and assign affiliations to authors.

In the case of a layout of the second type (Fig. 3), each author is already assigned to its affiliation by being placed in the same zone. It is therefore enough to detect author name, affiliation and email address. We assume the first line of such a zone is the author name, email is detected based on regular expressions,

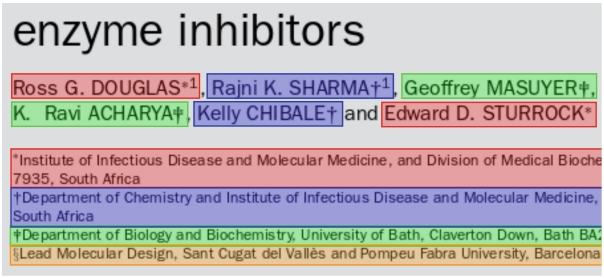

Fig. 2. An example fragment of a page from a scientific publication with authors and affiliations zones. In this case the relations author-affiliation (coded with colors) can be determined with the use of upper indexes.

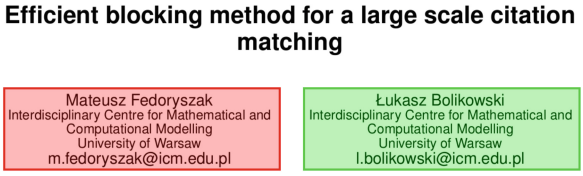

Fig. 3. An example fragment of a page from a scientific publication with authors and affiliations zones. In this case the relations author-affiliation can be determined using the distance and other geometric features of the text.

and the rest is treated as the affiliation string. In the future we plan to implement this step using a supervised token classifier.

3.4 Affiliation Parsing

The goal of affiliation parsing is to recognize affiliation string fragments related to institution, address and country. Additionally, country names are decorated with their ISO codes. Figure 4 shows an example of a parsed affiliation string.

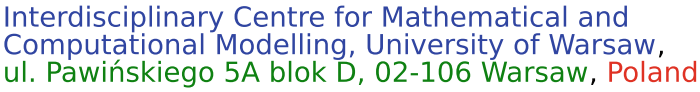

Fig. 4. An example of a parsed affiliation string. Colors mark fragments related to institution, address and country.

Affiliation parser uses Conditional Random Fields classifier and is built on top of GRMM and MALLET packages [10]. First affiliation string is tokenized, then each token is classified as *institution*, *address*, *country* or *other*, and finally neighbouring tokens with the same label are concatenated. The main feature used by token classifier is the classified word itself. Additional features are all binary: whether the token is a number, whether it is all uppercase/lowercase

word, whether it is a lowercase word that starts with an uppercase letter, whether the token is contained by dictionaries of countries or words commonly appearing in institutions or addresses. Additionally, the token's feature vector contains not only features of the token itself, but also features of two preceding and two following tokens.

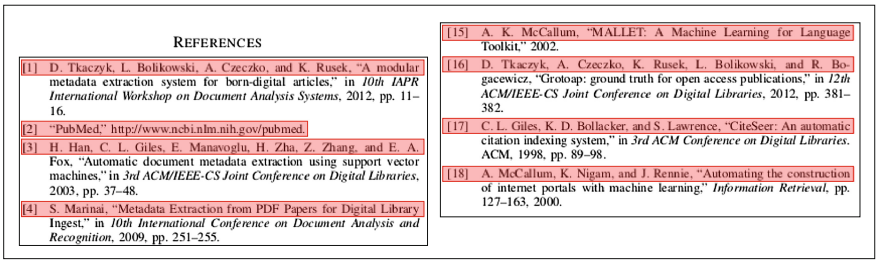

Fig. 5. A fragment of the references section of an article. Marked lines are the first lines of their references. After detecting these lines, the references section content can be easily split to form consecutive references strings.

3.5 References Extraction

References zones detected by content classifiers 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:

```
[fontsize=\small]
(
  <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. 5. To achieve this, we transform all lines to feature vectors and cluster them

into two sets. 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. As initial centroids we set the first line's feature vector and a vector with the largest distance to the first one. We use 5 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).

3.6 Reference 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* (author's fullname), *title*, *source* (journal or conference name), *volume*, *issue*, *pages*, *year* and *DOI*. An example of a parsed reference is shown in Fig. 6.

[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. 6. An example of a bibliographic reference with various metadata information highlighted using different colors, these are in order: *author*, *title*, *journal*, *volume*, *issue*, *pages* and *year*.

First the reference strings are 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 the classifier that assigns labels to reference string tokens. For better performance, the classifier uses slightly more detailed set of 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 [10].

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, eg. digits or lowercase/uppercase letters.

– Others check whether the token is a particular character (eg. 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, eg. 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 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.

Since the dataset used for training the token classifier does not contain enough references with DOI, the classifier is not responsible for extracting this information. DOI is recognized separately by matching a regular expression against the citation string.

Finally, the type of the reference (journal paper, conference proceedings or technical report) is detected by searching for specific keywords in the reference string.

4 Semantic Publishing Challenge of ESWC 2015

CERMINE system participated in Task 2 of Semantic Publishing Challenge of ESWC 2015 conference. The system is able to extract data sufficient for answering queries related to affiliations and citations (the first 5 queries out of 10 total, Q2.1 — Q2.5).

Solving queries Q2.1 (Affiliations in a paper) and Q2.2 (Papers from a country) relies on the following system features: document title extraction, authors and affiliations extraction, establishing relations author — affiliations, detecting country in the affiliation string.

Solving queries Q2.3 (Cited Works), Q2.4 (Recent Cited Works) and Q2.5 (Cited Journal Papers) relies on the following system features: extracting citations from a document, detecting DOI, title and year in the citation string, recognizing the type of a citation.

The following changes were made to the system in order to prepare it for the challenge:

– An additional step for generating the LOD dataset in RDF format was added to the original extraction workflow.
– The metadata classifier was retrained on a slightly extended set of documents, with the addition of documents of ACM layout.
– Heuristics for extracting authors and affiliations from hybrid zones (the second type described in Sect. 3.3, shown in Fig. 3) were added.

Fig. 7. The results of the evaluation of CERMINE in the SemPub Challenge. The figure shows mean precision, recall and F-score values for queries Q2.1 — Q2.5.

- Extracting DOI from reference strings based on regular expressions was implemented.
- Additional step for recognizing the type of the reference was added.

Originally both zone classifiers were trained with the use of a set of 2,551 documents randomly chosen from GROTOAP2 dataset [12]. Since GROTOAP2 was built using PMC resources, the dataset does not contain documents of ACM layout. For the purpose of the challenge, additional set of 165 ACM documents was manually chosen from computer science conferences and manually labelled. The combined set of 2,716 documents was used to retrain the metadata classifier.

Affiliation dataset used for affiliation parser training contains 8,267 parsed affiliations from PMC resources. For reference parser training we used CiteSeer [8], Cora-ref [9] and PMC resources combined together into a set of 4,000 references.

The LOD dataset generated by the workflow contains currently only the information needed to answer the challenge queries. More precisely, the dataset contains the following resources: volumes, documents, authors, affiliations (representing the relations between the document, author and organization), countries and citations. For all properties we use Dublin Core [1] and vCard [6] ontologies.

During the challenge 5 queries for each query type (50 single queries in total) were executed on the generated LOD dataset and the results were compared with the gold standard. The comparison was done after some normalization of the output and partial matches were also taken into account. For each query precision and recall were measured.

Figure 7 shows the mean scores of CERMINE for each query type Q2.1 — Q2.5. Since the generated LOD dataset did not contain any information supporting solving queries Q2.6 — Q2.10, all the scores for these queries were equal to 0. Table 1 shows the average precision, recall and F-score achieved by CERMINE over all queries, as well as only over the supported queries Q2.1 — Q2.5.

Table 1. The results of the evaluation of CERMINE in the SemPubl Challenge. The table lists the average precision, recall and F-score over all queries, as well as only for the first 5 queries. The scores for queries Q2.6 — Q2.10 were all equal to 0.

Queries	Q2.1 — Q2.10	Q2.1 — Q2.5
Precision	36.9 %	73.8 %
Recall	41.7 %	83.4 %
F-score	38.1 %	76.2 %

5 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. The modular architecture makes CERMINE flexible and easily maintainable.

CERMINE was designed as a universal solution, and therefore is able to handle a vast variety of potential publication layouts reasonably well, instead of being perfect in processing a limited number of document layouts only. This was achieved by employing supervised and unsupervised machine-learning algorithms trained on large, diverse datasets.

The system is open source and available online at http://cermine.ceon.pl.

References

1. Dublin Core. http://dublincore.org/
2. iText. http://itextpdf.com/
3. NLM. http://dtd.nlm.nih.gov/archiving/
4. PdfMiner. http://www.unixuser.org/~euske/python/pdfminer/
5. PubMed. http://www.ncbi.nlm.nih.gov/pubmed
6. vCard. http://www.w3.org/TR/vcard-rdf/
7. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. ACM TIST **2**(3), 27 (2011)
8. 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)
9. McCallum, A., Nigam, K., Rennie, J.: Automating the construction of internet portals with machine learning. Inf. Retrieval **3**, 127–163 (2000)
10. McCallum, A.K.: MALLET: A Machine Learning for Language Toolkit (2002)
11. O'Gorman, L.: The document spectrum for page layout analysis. IEEE Trans. Pattern Anal. Mach. Intell. **15**(11), 1162–1173 (1993)
12. Tkaczyk, D., Szostek, P., Bolikowski, L.: GROTOAP2 - the methodology of creating a large ground truth dataset of scientific articles. D-Lib Magazine (2014)
13. Tkaczyk, D., Szostek, P., Fedoryszak, M., Dendek, P.J., Bolikowski, L.: CERMINE: automatic extraction of structured metadata from scientific literature. Int. J. Doc. Anal. Recogn. (IJDAR), 1–19 (2015). http://dx.doi.org/10.1007/s10032-015-0249-8. doi:10.1007/s10032-015-0249-8
14. Tkaczyk, D., et al.: Cermine: Cermine 1.6 (2015). http://dx.doi.org/10.5281/zenodo.17594