STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXII**, Number 2, 2017
DOI: 10.24193/subbi.2017.2.01

A HYBRID APPROACH FOR SCHOLARLY INFORMATION EXTRACTION

ZALÁN BODÓ AND LEHEL CSATÓ

Abstract. Metadata extraction from documents forms an essential part of web or desktop search systems. Similarly, digital libraries that index scholarly literature require to find and extract the title, the list of authors and other publication-related information from an article. We present a hybrid approach for metadata extraction, combining classification and clustering to extract the desired information without the need of a conventional labeled dataset for training. An important asset of the proposed method is that the resulting clustering parameters can be used in other problems, e.g. document layout analysis.

1. Introduction

Since its inception in the 1970s, information retrieval is heavily used in several domains of science, not to mention its indispensable everyday use as back-ends for search engines. Search engines collect the available documents and process the various formats to (a) extract information like title, abstract, and authors of a publication and (b) to rank these documents according to the query posed by the user. In this article we focus on document processing and authorship, respectively title extraction; we call this information metadata. The metadata for a scientific publication is of great importance for digital libraries, the ranking algorithms perform best when complete and unambiguous information is provided.

We present a generic hybrid method that considers the specifics of the metadata to be extracted, and optimizes a clustering procedure that concatenates text chunks from within a document. A beneficial by-product of this method is that the resulting clustering is amenable for other domains, such as document layout analysis [3]. We also mention that our system does not need labeled

Received by the editors: June 19, 2017.

2010 *Mathematics Subject Classification.* 62H30, 68P20.

1998 *CR Categories and Descriptors.* I.2.6 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE – *Learning*; I.5.3 [**Computing Methodologies**]: PATTERN RECOGNITION – *Clustering*.

Key words and phrases. information extraction, metadata, machine learning.

data as input, but only a set of PDF documents along with a metadata database, the labeling procedure is embedded into the training phase.

The paper is structured as follows: Section 2 introduces the problem of metadata extraction and enumerates popular PDF extraction tools, and Section 3 describes the proposed algorithm for metadata extraction. In Section 4 the features used for representing the text segments are described, and Section 5 concludes the paper by presenting the experiments and discussing the results.

2. Metadata extraction from scholarly articles

Metadata extraction is generally viewed as a classification problem, where specific text segments are needed to be identified and extracted from a document. That is, if we consider the text chunks extracted from a PDF document, these have to be labeled as being part of the title, of the author list, of the *other* category, etc. Hence, the problem can be viewed as a supervised learning task: given a set of labeled examples, learn the mapping from the data to labels. This can be done either by a classifier (e.g. support vector machine [7]) or by a structured learning method (e.g. conditional random field [14]). In either case labeled examples or sequences are needed in order to train the learning model chosen. However, the problem can also be approached by unsupervised learning: without using any labeled data, find the cohesive text segments of the page, then label these using for example some kind of rule set [8]. The works [6, 10] give an excellent overview of the available methods and tools.

In this paper we consider the problem as a supervised learning task, but we also make use of clustering methods to find the cohesive text segments, that will be used for learning.

Although scholarly articles can be found in a wide variety of file formats on the Internet, the *Portable Document Format* is without doubt the most popular one. Therefore, it is sufficient to consider this format for metadata extraction.

The most popular tools used for obtaining the text fragments from PDF documents—for Java, C# and Python—are PDFBox[1], iText[2] and PDFMiner[3] [1, 4, 16]. All of these libraries can extract text along with font and positional information from a document, however, the implementations and functionalities—obviously—differ. Thus, some of them can recognize and separate ligatures, others cannot, they return different reading orders for the text

[1] `http://pdfbox.apache.org/`

[2] `http://itextpdf.com/`

[3] `http://www.unixuser.org/~euske/python/pdfminer/`

chunks, etc.[4] Figure 1(a) shows the text chunks returned by PDFMiner for a test document.

3. A hybrid approach for metadata extraction

In order to use the described method, a large training set of scholarly articles is required with metadata information attached to them, e.g the title of the paper, author names, journal or conference proceedings the article appeared in, etc., depending on what kind of metadata is going to be extracted, in textual format. This can be obtained by using a specialized crawler, which seeks for scientific articles on specific websites, and finds the associated metadata. This is by itself a complex task to perform, therefore the setup of such a system is not detailed here. Another possibility is to use a digital library, from where the documents and metadata can be obtained in a more straightforward manner. One such digital library is CiteSeerX [5, 17], which offers an OAI collection for metadata harvesting.[5] Other approaches may include the utilization of large research article databases such as the ACL Anthology[6], PubMed Central Open Access Subset[7] or the arXiv e-Print Archive[8]. Online and open bibliography websites such as DBLP[9] or the Collection of Computer Science Bibliographies[10] also offer a huge amount of bibliographic data. Combined with a web search engine with high coverage, one can obtain a large collection of articles and associated metadata from various journals and conference proceedings. In order to generalize well, it is of central importance to train the metadata extraction system using a large variety of article formats.

PDF extraction tools return the extracted text as separate text chunks or segments along with positional and font information. We consider the problem of information extraction as a two-step procedure: (a) cluster the segments to find the cohesive parts, e.g. the title of the article, (b) use the output of the clustering as input for a supervised learning method. We do not require

[4] Using 100 random articles as a test set, it resulted that the fastest is iText, PDFBox is about twice as slow, while PDFMiner is the slowest of the three libraries, slower than iText by a factor of 7. The tested versions were 7.0.1, 2.0.3 and 20140328, respectively.

[5] CiteSeerX also employs a metadata extraction system to extract these information from the crawled files, hence it might seem odd to use the output of a metadata extraction system as input for training another metadata extraction system. However, a similar wrapper method, where the output is fed back as input, called *self-training*, is a common approach in semi-supervised learning to strengthen the confidence of the underlying classifier [18].

[6] `http://acl-arc.comp.nus.edu.sg/`

[7] `ftp://ftp.ncbi.nlm.nih.gov/pub/pmc/`

[8] `https://arxiv.org/`

[9] `http://dblp.uni-trier.de/`

[10] `http://liinwww.ira.uka.de/bibliography/`

Algorithm 1 Finding the clustering parameters

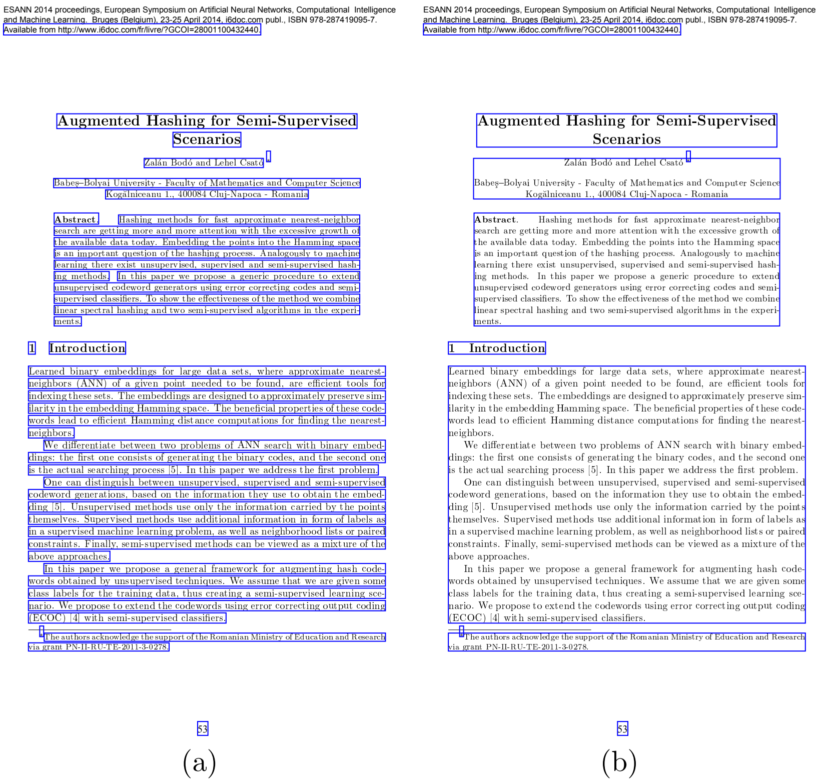
```
Choose similarity threshold $t$
while best clustering parameters $P$ are found do
    $P \leftarrow$ next parameter set
    Perform clustering using $P$
    Count matches using threshold $t$
    Evaluate $P$: #matches/#all cases
end while
```

a conventional labeled dataset for training, but only a set of PDF documents with a metadata database.

Our method searches for the best clustering of the extracted text chunks, such that to maximize the number of matches between the metadata and the obtained text segments. A match is found if the similarity between the metadata and the text segment does exceed a given threshold, and this happens exactly once. No match or multiple matches are equivalently considered as no matches. The proposed method has a number of parameters to set, including the clustering algorithm and therefore its parameters, the similarity measure and threshold. However, since we are working with textual data, we recommend using the bag-of-n-grams representation with raw frequency weighting scheme, considering the relative shortness of the text segments, and the cosine similarity measure [15, 13]. Algorithm 1 summarizes the described process. Searching for the optimal clustering parameters can be done using either randomized or grid search.

As possible benefits of the proposed procedure we enumerate the following:

- Constructing a labeled dataset for metadata extraction is a costly process. The approach presented in this paper does not need a conventional labeled dataset for training, but a set of PDF files along with a metadata database containing the metadata to be found/extracted, which can be much more less expensive to produce.
- The clustering procedure (i.e. the clustering parameters) can be used in another system, that requires to determine cohesive text segments in a document, for example in document layout analysis [3].
- The output text segments can be used in either a classification or a sequence tagging algorithm, thus, the learning method can use font and position-related features. Fewer text segments can increase the performance of the learning method.
- Using a measure of central tendency, the font name, font size, font style are determined for the entire segment, thus reducing the probability of misguiding the classifier—when different font styles or sizes



ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Augmented Hashing for Semi-Supervised Scenarios

Zalán Bodó and Lehel Csató [*]

Babeş–Bolyai University - Faculty of Mathematics and Computer Science
Kogălniceanu 1., 400084 Cluj-Napoca - Romania

Abstract. Hashing methods for fast approximate nearest-neighbor search are getting more and more attention with the excessive growth of the available data today. Embedding the points into the Hamming space is an important question of the hashing process. Analogously to machine learning there exist unsupervised, supervised and semi-supervised hashing methods. In this paper we propose a generic procedure to extend unsupervised codeword generators using error correcting codes and semi-supervised classifiers. To show the effectiveness of the method we combine linear spectral hashing and two semi-supervised algorithms in the experiments.

1 Introduction

Learned binary embeddings for large data sets, where approximate nearest-neighbors (ANN) of a given point needed to be found, are efficient tools for indexing these sets. The embeddings are designed to approximately preserve similarity in the embedding Hamming space. The beneficial properties of these codewords lead to efficient Hamming distance computations for finding the nearest-neighbors.

We differentiate between two problems of ANN search with binary embeddings: the first one consists of generating the binary codes, and the second one is the actual searching process [5]. In this paper we address the first problem.

One can distinguish between unsupervised, supervised and semi-supervised codeword generations, based on the information they use to obtain the embedding [5]. Unsupervised methods use only the information carried by the points themselves. Supervised methods use additional information in form of labels as in a supervised machine learning problem, as well as neighborhood lists or paired constraints. Finally, semi-supervised methods can be viewed as a mixture of the above approaches.

In this paper we propose a general framework for augmenting hash codewords obtained by unsupervised techniques. We assume that we are given some class labels for the training data, thus creating a semi-supervised learning scenario. We propose to extend the codewords using error correcting output coding (ECOC) [4] with semi-supervised classifiers.

[*] The authors acknowledge the support of the Romanian Ministry of Education and Research via grant PN-II-RU-TE-2011-3-0278.

53

(a)

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Augmented Hashing for Semi-Supervised Scenarios

Zalán Bodó and Lehel Csató [*]

Babeş–Bolyai University - Faculty of Mathematics and Computer Science
Kogălniceanu 1., 400084 Cluj-Napoca - Romania

Abstract. Hashing methods for fast approximate nearest-neighbor search are getting more and more attention with the excessive growth of the available data today. Embedding the points into the Hamming space is an important question of the hashing process. Analogously to machine learning there exist unsupervised, supervised and semi-supervised hashing methods. In this paper we propose a generic procedure to extend unsupervised codeword generators using error correcting codes and semi-supervised classifiers. To show the effectiveness of the method we combine linear spectral hashing and two semi-supervised algorithms in the experiments.

1 Introduction

Learned binary embeddings for large data sets, where approximate nearest-neighbors (ANN) of a given point needed to be found, are efficient tools for indexing these sets. The embeddings are designed to approximately preserve similarity in the embedding Hamming space. The beneficial properties of these codewords lead to efficient Hamming distance computations for finding the nearest-neighbors.

We differentiate between two problems of ANN search with binary embeddings: the first one consists of generating the binary codes, and the second one is the actual searching process [5]. In this paper we address the first problem.

One can distinguish between unsupervised, supervised and semi-supervised codeword generations, based on the information they use to obtain the embedding [5]. Unsupervised methods use only the information carried by the points themselves. Supervised methods use additional information in form of labels as in a supervised machine learning problem, as well as neighborhood lists or paired constraints. Finally, semi-supervised methods can be viewed as a mixture of the above approaches.

In this paper we propose a general framework for augmenting hash codewords obtained by unsupervised techniques. We assume that we are given some class labels for the training data, thus creating a semi-supervised learning scenario. We propose to extend the codewords using error correcting output coding (ECOC) [4] with semi-supervised classifiers.

[*] The authors acknowledge the support of the Romanian Ministry of Education and Research via grant PN-II-RU-TE-2011-3-0278.

53

(b)

FIGURE 1. Results of text chunk extraction using PDFMiner (a) before and (b) after clustering.

are assigned for consecutive text chunks actually belonging to the same segment.[11]

Figure 1(b) shows the result obtained by clustering the text chunks extracted from a PDF document using PDFMiner. The parameters of the clustering method were determined using the proposed algorithm.

4. Clustering and building the feature vectors

Since PDFMiner returns concatenated text chunks from the same line (see Figure 1), the clustering is performed in vertical direction only. For clustering the text chunks/segments, a distance measure between the objects is needed,

[11] However, we mention that the clustering method is faced with the same challenge during the clustering process.

for which the following metric was used:

$$\begin{aligned} d(x,z) &= \min(\min(d_1(x,z), d_2(x,z)), d_3(x,z) + d_4(x,z)) \\ &\quad /(lineStretch \cdot minSize) + sizeFactor \cdot d_5(x,z) \end{aligned} \tag{1}$$

where d_1, d_2, d_3 and d_4 return the distances between the bottom left and top right, top right and bottom left, bottom left and bottom left, and top right and top right y-coordinates of the bounding boxes, respectively. The parameter $minSize$ gives the minimum of the most frequent font sizes of the two text segments, while d_5 is the absolute value of the font size difference. In case the majority font styles and font names differ between the two segments, the distance is multiplied by a $styleFactor$ and $fontFactor$ parameter, respectively. These parameters of the distance function, along with $lineStretch$ and $sizeFactor$, were determined using cross-validation.

For representing text chunks for the supervised learning phase we use the following 3 feature categories:

(a) distances, sizes;
(b) regular expression-based features;
(c) dictionary-based features.

Distance and size features include the distances between the previous and trailing text chunks (i.e. their bounding boxes), vertical positions of the bounding box, index of the text chunk, text length, number of words, font sizes (current, before and after, most frequent in the segment), font style (regular, bold, italic). Regular expressions-based features include the ratios of uppercase letters and terms (current, before and after), presence of email and URL addresses, ratio of numbers and special characters (non-word characters). Additionally, a database of first and last names[12] was used for calculating the ratio of names found in a text chunk.

5. Experiments and discussion

5.1. The dataset. The training and test data were obtained from CiteSeerX by retrieving 5000 scholarly articles from it, using the CiteSeerX OAI collection via the harvest URL.[13]. The dataset was compiled between September 5 and 7, 2016, using selective harvesting with datestamps of 01.01.2005 and 01.01.2016. We stored the metadata of an article only if the *source* field was valid, and stopped when the number of 5000 articles was reached. The downloaded files were processed using PDFMiner omitting the erroneous (e.g. non-PDF) documents, resulting in a total of 4217 articles. This collection was split randomly into 3 sets: C_1 with 1000 documents, C_2 also with 1000 documents,

[12] `https://github.com/SeerLabs/CiteSeerX`

[13] `http://citeseerx.ist.psu.edu/oai2`

```
{...
"http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.5389": {
  "source": "http://www.csie.ntu.edu.tw/~cjlin/papers/svmprob/svmprob.pdf",
  "author": [
    "Ting-fan Wu",
    " Chih-Jen Lin",
    " Ruby C. Weng"
  ],
  "title": "Probability Estimates for Multi-class Classification by
           Pairwise Coupling"
  }
...}
```

FIGURE 2. Data stored for an article in our CiteSeerX-based dataset.

Matching errors	Cluster distance threshold								
Similarity threshold	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
0.6	124	111	104	99	92	**88**	89	**88**	89
0.65	91	84	82	80	77	**75**	77	78	81
0.7	79	75	74	71	71	**69**	72	73	75
0.75	91	88	**86**	**86**	87	87	89	90	92
0.8	134	122	118	115	**113**	**113**	116	116	118

TABLE 1. Matching errors obtained on set C_1 for titles by varying the similarity threshold from 0.6 to 0.8 by 0.05 (rows) and the cluster distance threshold from 0.4 to 2.0 by 0.2 (columns).

and C_3 with the remaining 2217 articles. C_1 was used for determining the best clustering parameters, C_2 was the training set, while the system was tested on C_3.

The dataset, which by its nature contains errors and inaccuracies, can be freely downloaded and used for further research.[14] The data is availabe in JSON format, where the key is the CiteSeerX identifier/URL of the article, and each item is described by three fields, *title*, *author* and *source*, the latter containing the URL to the downloadable file. Figure 2 shows the description of an article in this dataset.

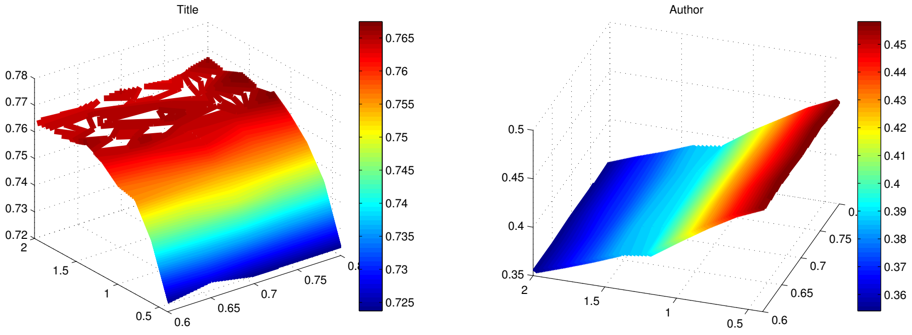
5.2. **Experimental results.** Because our entire system was implemented in Python, despite its disadvantages, such as relative slowness and omission of certain chunks at extraction (see Figure 1(a)), the PDFMiner package was used.

For clustering the text chunks single linkage hierarchical clustering was used with a parametrized distance function, taking into account the distances between the chunks, the font names, sizes and styles, as described in Section 4.

[14] `http://www.cs.ubbcluj.ro/~zbodo/citeseerx4217.html`

Matching errors	Cluster distance threshold								
Similarity threshold	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
0.6	**274**	290	299	317	332	347	362	371	385
0.65	**297**	319	332	353	379	392	409	420	434
0.7	**337**	358	372	391	418	433	451	467	479
0.75	**389**	412	425	446	475	491	506	520	530
0.8	**438**	461	468	487	515	530	544	555	567

Table 2. Matching errors obtained on set C_1 for authors, by varying the similarity threshold from 0.6 to 0.8 by 0.05 (rows) and the cluster distance threshold from 0.4 to 2.0 by 0.2 (columns).



(a)

(b)

Figure 3. Errors obtained on set C_3 for (a) titles and (b) authors, using the same search grid as in Tables 1–2.

From the design of the distance metric it follows that the optimal threshold should be somewhere around 1, therefore we performed a grid search around this value.

The applied method for metadata extraction is similar to the one presented in [9], but for determining the text segments, we used the method described in Section 3. Our system currently extracts only the title and the authors of the article, without segmenting the author names. The feature vectors—including dictionary-based features, regular expressions, positional and font information—are constructed for each segment, and random forest classifiers

Accuracy	Title	Authors
SVMHeaderParse with PDFBox	0.6982	0.4713
SVMHeaderParse with PDFMiner	0.6414	0.4848
GROBID	0.7780	0.6337
Our method	0.7631	0.4091

TABLE 3. Accuracy results using C_3 for testing.

[2] are trained on these data. Two sets of 100 random decision trees were trained using the *scikit-learn* library[15], one for each class.

The distance threshold of the hierarchical clustering and the similarity threshold for finding the corresponding metadata influence each other, therefore we performed a joint search for the optimal values. In Tables 1–2 the number of matching errors obtained on set C_1 are shown, varying the clustering distance and similarity threshold. For these parameter combinations the metadata extraction system was trained on set C_2 and evaluated on C_3: Figure 3 shows the errors obtained. At evaluation we used a testing similarity threshold of 0.9 for checking whether the corresponding metada was found. Similarity was measured using cosine similarity, representing the texts as bag-of-words weighted by the frequencies. The author names were considered as one object.

We selected the best parameters by standardizing the rows of the matching error matrices, summing the two and finding the minimum of each row. Other methods for finding the optimal joint parameters can be applied as well, for example using an importance weighted linear combination of the matching errors for the different categories. Thus, for example using a similarity threshold 0.7 we obtained the best cluster threshold of 1.0. For these values we get accuracies of 0.7631 and 0.4091 for titles and respectively authors, using the above-mentioned similarity threshold of 0.9 in testing.

We compared our approach to other existing methods, as shown in Table 3. However, we were faced with the following difficulties when testing other systems. First, unlike our approach, existing systems [10] use supervised learning for training—either a classifier or a structured learning method—and use distinct sets of classes. Our solution, however, does not need labeled data, but only a set of PDF files along with a metadata database (see Figure 2), the labeling procedure is embedded into the training phase. Second, existing systems were primarily designed to use them just as they were trained, therefore re-training is not well-documented and difficult to perform. Hence, we were

[15] `http://scikit-learn.org/`

not able train these using set C_2, but only tested them on C_3. SVMHeaderParse is the metadata extraction module used by CiteSeerX implementing the method described in [7]. For text extraction it uses PDFBox, but since this component can be easily replaced, we also tested it using PDFMiner. GROBID[16] (GeneRation Of BIbliographical Data, version 0.4.1) is a complex metadata extraction tool for header metadata and bibliographical extractions. GROBID uses pdftoxml[17] for content and layout extraction and conditional random fields for learning [11, 12]. As the results in Table 3 show, our method performs well on title extraction, getting almost the same accuracies as GROBID, which obtained the best overall results in the experiments of [10]. For author extraction our system achieved the lowest accuracy results, thus a careful examination of the obtained results is required to be able to improve on the performance.

5.3. **Discussion.** In this paper we described a hybrid metadata extraction system, that uses clustering to identify cohesive text segments, after which, based on the features representing a segment, supervised learning is used to automatically find parts containing metadata information. The method described does not need a conventional labeled dataset for training, but only a set of PDF documents along with a metadata database. We also compiled a small dataset from CiteSeerX's database and used it to train and evaluate our method.

From the experiments it can be seen that finding titles is easier—this can be argued by the fact that the most important information about an article is its title, which always constitutes the most accentuated part of the title page. The relatively low accuracy obtained for authors can be explained by the noise in the training data, as well as by the fact that the author list is often broken up by additional information, e.g. affiliation, which makes the recognition process more difficult. Using less noisy training and test data our system achieves accuracies up to 90% and 70% for titles and authors, respectively.

The obtained results show that matching errors vary differently for distinct metadata categories, therefore a sound methodology is needed how to select the overall optimal parameters of the clustering method and possibly the similarity threshold too. Future investigations also include the application and testing of structured prediction models in metadata extraction using the proposed method.

[16] `https://github.com/kermitt2/grobid`

[17] `https://github.com/eliask/pdf2xml`

Acknowledgements. This work was supported by a grant of the Romanian National Authority for Scientific Research, CNDI–UEFISCDI, project number PN-II-PT-PCCA-2011-3.2-0895.

References

[1] J. Beel, S. Langer, M. Genzmehr, and C. Müller. Docear's PDF inspector: title extraction from PDF files. In *JCDL*, pages 443–444. ACM, 2013.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5, 2001.

[3] T. M. Breuel. High performance document layout analysis. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 209–218, 2003.

[4] B. H. Butt, M. Rafi, A. Jamal, R. S. U. Rehman, S. M. Z. Alam, and M. B. Alam. Classification of research citations (CRC). In *CLBib@ISSI*, volume 1384 of *CEUR Workshop Proceedings*, pages 18–27. CEUR-WS.org, 2015.

[5] C. Caragea, J. Wu, A. M. Ciobanu, K. Williams, J. P. F. Ramírez, H.-H. Chen, Z. Wu, and C. L. Giles. CiteseerX: A scholarly big dataset. In *ECIR*, volume 8416 of *Lecture Notes in Computer Science*, pages 311–322. Springer, 2014.

[6] M. Granitzer, M. Hristakeva, K. Jack, and R. Knight. A comparison of metadata extraction techniques for crowdsourced bibliographic metadata management. In *SAC*, pages 962–964. ACM, 2012.

[7] H. Han, C. L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E. A. Fox. Automatic document metadata extraction using support vector machines. In *JCDL*, pages 37–48. IEEE Computer Society, 2003.

[8] A. Ivanyukovich and M. Marchese. Unsupervised metadata extraction in scientific digital libraries using a-priori domain-specific knowledge. In *SWAP*, volume 201 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[9] R. Kern, K. Jack, M. Hristakeva, and M. Granitzer. Teambeam - meta-data extraction from scientific literature. *D-Lib Magazine*, 18(7/8), 2012.

[10] M. Lipinski, K. Yao, C. Breitinger, J. Beel, and B. Gipp. Evaluation of header metadata extraction approaches and tools for scientific PDF documents. In *JCDL*, pages 385–386. ACM, 2013.

[11] P. Lopez. Grobid: Combining automatic bibliographic data recognition and term extraction for scholarship publications. In *International Conference on Theory and Practice of Digital Libraries*, pages 473–474. Springer, 2009.

[12] P. Lopez and L. Romary. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 248–251. Association for Computational Linguistics, 2010.

[13] C. D. Manning, H. Schütze, and P. Raghavan. *Introduction to information retrieval.* Cambridge University Press, 2008.

[14] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL*, pages 329–336, 2004.

[15] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[16] J. Wu, J. Killian, H. Yang, K. Williams, S. R. Choudhury, S. Tuarob, C. Caragea, and C. L. Giles. PDFMEF: A multi-entity knowledge extraction framework for scholarly documents and semantic search. In *K-CAP*, pages 13:1–13:8. ACM, 2015.

[17] J. Wu, K. M. Williams, H.-H. Chen, M. Khabsa, C. Caragea, S. Tuarob, A. Ororbia, D. Jordan, P. Mitra, and C. L. Giles. CiteseerX: AI in a digital library search engine. *AI Magazine*, 36(3):35–48, 2015.

[18] X. Zhu. Semi-supervised learning literature survey. Technical Report TR 1530, University of Wisconsin, 2005.

Faculty of Mathematics and Computer Science, Babeş–Bolyai University, Cluj-Napoca, Romania

E-mail address: {zbodo,lehel.csato}@cs.ubbcluj.ro