Workflow of metadata extraction from retro-born-digital documents

Dominika Tkaczyk and Łukasz Bolikowski

Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, ul. Pawińskiego 5A blok D, 02-106 Warsaw, Poland

Abstract. In this work-in-progress report we propose a workflow for metadata extraction from articles in a digital form. We decompose the problem into clearly defined sub-tasks and outline possible implementations of the sub-tasks. We report the progress of implementation and tests, and state future work.

Keywords: metadata extraction, page segmentation, zone classification, Hidden Markov Model

1 Introduction

Whenever a digital library acquires a document without metadata, or with metadata of poor quality, there is a need for extracting metadata from the content at hand. In this paper we focus on extracting metadata of scientific articles. Our goal is to extract as much information as possible, including: title, authors, affiliations, abstract, parsed bibliographic references, journal, volume, issue, pages, and year of publication. At this stage, we are not interested in inferring missing information based on the text of the document, such as: language of the document, keywords or categories (unless they are explicitly listed in the front matter).

The problem of extracting metadata and content from a document is well-studied in the literature. We shall follow the nomenclature presented by Sojka [10]. Older approaches assume that an image of a document is available on input, and execute full digitisation from bitmap image. This was a reasonable assumption in the past, when documents were scanned and retro-digitised. For example, the Medical Article Records System (MARS) [4] works on document scans in the form of TIFF images.

Nowadays, we see more and more retro-born-digital documents, and there is no need to recognise individual characters. This difference has an impact on both the workflow and the performance of metadata extraction process. For example, Cui and Chen [5] employ a Hidden Markov Model to extract metadata from PDF documents, while page segmentation and text extraction is done by Pdftohtml, a third-party open-source tool. Giles *et al.* Marinai [7] uses JPedal package for extracting characters from PDF, performs rule-based page segmentation, and employs neural classifier for zone classification.

2 Metadata extraction workflow

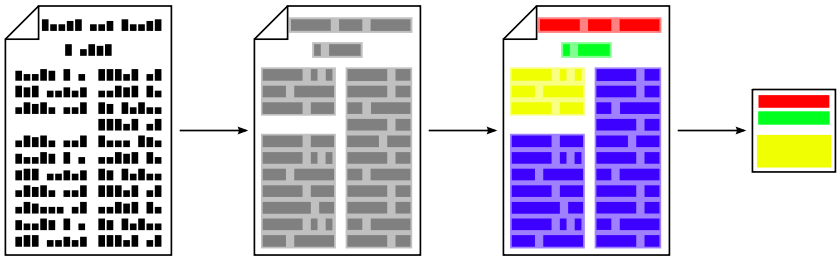

Fig. 1. From a PDF file to a metadata record. The leftmost document contains characters with bounding boxes, as extracted from the PDF file. After page segmentation, the characters are clustered into words, lines and zones, as shown in the second document. Next, the zones are classified as title, authors, affiliations, abstract, body, page number, etc. A metadata record (first to the right) is built based on the labelled zones.

In this section we describe all steps of the metadata extraction process. The process begins with an electronic document (our current implementation supports only PDF format) and its output is the metadata extracted from it. There are three main stages of the process:

1. Building a tree structure that stores the document's content, described in subsections 2.1 and 2.2.
2. Analysing and enhancing the document's content based on its structure, described in subsections 2.3, 2.4 and 2.5.
3. Generating the document's metadata based on enhanced content, described in subsection 2.6.

2.1 Character extraction

The purpose of the first step of the process is to build an initial, flat structure storing the document's content. During character extraction an electronic document is processed. The output of character extraction is a list of document's pages, each of which contains a set of individual characters with their bounding boxes.

Our current implementation of character extraction is based on iText library [1] and is able to process documents in PDF format. iText is used to iterate over PDF's text-showing operators found in the document. During the iteration, we extract individual characters, their position on page, their width and height, and gather them to build initial document's structure. It is worth to notice that the character boxes we extract are in fact not the smallest rectangles enclosing characters, as iText does not provide such accurate information. In particular, all characters of the same font and size will have the same height.

Moreover, extracted character widths can differ from the exact values depending on characters and fonts used.

Apart from iText, we also considered using another open source PDF library PDFBox [3]. Our final choice was iText, due to its convenient and well organized API and satisfying character extraction results.

2.2 Page segmentation

In this step we group individual characters into words, lines and zones to build a tree structure representing document's content. After page segmentation the document consists of pages, each page consists of zones, each zone consists of text lines, each line consists of words, and finally each word consists of individual characters. All zones, lines and words can be described by their content, position on the page, width and height.

Our first implementation of page segmentation was a top-down approach based on X-Y cut algorithm [8]. In this solution, the document's page is recursively divided into rectangular blocks by horizontal or vertical cuts.

In the future we plan to replace the first solution with an implementation of a bottom-up Docstrum algorithm [9]. In this approach, the distances and angles between nearest-neighbour pairs of individual characters on the page are analysed, which allows to estimate the text line orientation angle, and also in-line and between-line spacing. Based on these information, we can group individual characters into words and lines, and finally group lines into zones.

In contrast to X-Y cut, Docstrum is independent from text line orientation and text spacing used in the document. Thus, we expect to get better results from it.

2.3 Zone classification

Different zones can have different meaning: a zone can represent document's author, title, abstract, etc. To classify zones means to associate them with labels from a predefined label set. Labels we use in our zone classifier are: *abstract*, *affiliation*, *author*, *body*, *footer*, *header* and *title*. Our implementation of the classifier is based on a Hidden Markov Model [6] with all probability information obtained from a training set.

The classifier processes sequences of all zones of a page sorted accordingly to their position on the page. Correct labels of zones are unknown, but each zone can be described by a vector of features calculated from zone's content, position and dimensions. We treat such sequences of zones with unknown labels as sequences of hidden states in a Hidden Markov Model. The vectors of features are messages emitted in every state. We use Viterbi algorithm to calculate the most probable states (labels) of a sequence of objects (zones) based on emitted messages (feature vectors).

To allow the Viterbi algorithm to perform its task, we have to calculate initial and transition state probability, and be able to calculate emission probability

for every feature vector. All probability information needed is obtained from a training set. Each training element has a form of a sequence of zones with known labels and vectors of features. Initial and transition probability can be calculated directly from the training set. To be able to calculate emission probability for every possible feature vector, we build a decision tree based on feature vectors of all training elements. The emission probability of a given feature vector can be calculated from those training elements, that are classified in the same leaf of the decision tree as given feature vector.

Our classifier uses 37 features to describe the document's zones. Some of the features refer to the zone's bounding box and its position on the page, e.g. zone's absolute and relative dimensions, horizontal and vertical position. We also used features related to the inner structure of the zone, e.g. absolute and relative number of text lines and words, mean text line height, width and position. Finally, some of the features are based on the text of the zone, e.g. the number of characters, digits, lowercase/uppercase letters, punctuation marks, etc.

We used documents obtained from MARG repository [2] for both training and test sets. The training set we used consisted of 317 elements and 1379 zones. The tests we performed on 1236 documents with 5359 zones gave the accuracy rate 95.5%.

It is worth to notice that apart from Hidden Markov Models there are many other supervised learning approaches available, e.g. Conditional Random Fields [11] or simple rule-based classiffiers. We chose Hidden Markov Models approach because of relatively simple algorithms needed, high maintainability of the solution and good quality of results.

2.4 Bibliographic references extraction

Extracting bibliographic references from a document is a first stage of references processing, which also includes references parsing and matching.

Our current bibliographic references extractor processes only the text content of a document and is based on simple character frequency heuristic. First, we select lines with a sufficiently high frequency of digits and punctuation. Next, we remove isolated lines and fill the gaps. Finally, extracted lines are concatenated and references split.

In the future we plan to implement a better bibliographic reference extractor, that makes use not only of the text content, but also of the document's tree structure constructed in previous steps. We believe that taking into account text positioning parameters, such as lines' positions in the document, the between-line spacing or line indentation will result in better bibliographic references lines detecting and grouping.

2.5 Bibliographic references parsing

To make further bibliographic references analysis (e.g. matching references with documents) possible, we have to parse extracted references and identify their

fragments containing author, title, journal, etc. Our implementation of bibliographic references parser is based on a Hidden Markov Model with all probability information obtained from a training set. Labels we use to tag fragments of references are: *author*, *title*, *journal*, *volume*, *series*, *number*, *publisher*, *location, edition*, *pages*, *url*, *year* and *content*.

A bibliographic reference can be represented as a sequence of tokens with unknown labels. Each token can be described by a vector of features calculated from its content. The sequence of tokens can be treated as a sequence of hidden states in a Hidden Markov Model. The vectors of features are messages emitted in every state. We use Viterbi algorithm to calculate the most probable sequence of states (token labels) based on emitted messages (feature vectors).

As in the case of zone classifier, all probability information needed by Viterbi algorithm is obtained from a training set. Training set consists of bibliographic references with tagged tokens. Initial and transition probability can be calculated directly from the training set, emission probability is calculated based on the decision tree constructed from training elements' feature vectors.

Our parser uses 48 features to describe citation's tokens. Some of the features measure relative number of particular character type, e.g. digits or uppercase letters. Other features check whether the token is a particular character (a comma, a dot, a quote, etc.) or a particular word ("and", "http", "vol", etc.). We also use features that are based on dictionaries built from the training set, e.g. a dictionary of cities or words commonly occuring in the journal title.

Both training and test sets were obtained from digital collections NUMDAM and CEDRAM. The training set we used consisted of 2318 bibliographic references. The tests we performed on 2532 references gave the accuracy rate 85.8% of correctly identified fragments of bibliographic references.

Apart from Hidden Markov Models, there are other supervised learning approaches available, e.g. Conditional Random Fields or template matching using regular expressions. As in the case of the zone classifier, we chose Hidden Markov Model approach due to its simplicity, flexibility and good quality results.

2.6 Metadata extraction

In the final step the metadata is extracted based on labelled zones, document structure and content. The final step hasn't been implemented yet.

3 Current status and future work

So far we have implemented and tested most of the metadata extraction steps described in the previous section. Our implementation of character extraction is based on iText library, for page segmentation we currently use X-Y cut algorithm and zone classifier is based on a Hidden Markov Model. We have also implemented bibliographic references processing: references extractor uses simple character frequency heuristic and parser is based on a Hidden Markov Model.

However, the metadata extraction process still needs some work. In the near future we plan to replace current implementation of the page segmenter with a version based on Docstrum algorithm. We are also going to implement a better bibliographic reference extractor, that makes use not only of the text content, but also of the document's tree structure. Also the final step of the process, the metadata extraction, is still to be implemented.

So far for testing page segmentation and zone classification implementations we have been using documents obtained from MARG repository. Unfortunately, MARG repository has its drawbacks: it contains only first pages of the documents and only a subset of all zones is included in the document's structure. In the future we plan to semi-manually construct a better test set. We hope that it will make our implementations and test results more reliable.

4 Summary

We have proposed a workflow for metadata extraction which is especially suitable for retro-born-digital documents, while still applicable in the case of full digitisation from bitmap images. We have reported our current implementation and testing efforts and stated future work.

5 Acknowledgements

This work is partly financed by the EuDML project, which is in turn partly financed by the European Union through its Competitiveness and Innovation Programme (Information and Communications Technologies Policy Support Programme, "Open access to scientific information", Grant Agreement no. 250,503). We would also like to thank the anonymous reviewers for their insightful comments.

References

1. iText, `http://itextpdf.com/`
2. MARG, `http://marg.nlm.nih.gov/`
3. PDFBox, `http://pdfbox.apache.org/`
4. Automating the production of bibliographic records for MEDLINE. Tech. rep. (2001)
5. Cui, B., Chen, X.: An improved hidden Markov model for literature metadata extraction. Advanced Intelligent Computing Theories and Applications pp. 205–212 (2010)
6. Hetzner, E.: A simple method for citation metadata extraction using Hidden Markov Models. In: JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries. pp. 280–284. ACM, New York, NY, USA (2008)
7. Marinai, S.: Metadata Extraction from PDF Papers for Digital Library Ingest. 2009 10th International Conference on Document Analysis and Recognition pp. 251–255 (2009)

8. Nagy, G., Seth, S., Viswanathan, M.: A prototype document image analysis system for technical journals. Computer 25(7), 10–22 (1992)
9. O'Gorman, L.: The document spectrum for page layout analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 15(11), 1162–1173 (1993)
10. Sojka, P.: An Experience with Building Digital Open Access Repository DML-CZ. In: Proceedings of CASLIN 2009. pp. 74–78 (2009)
11. Sutton, C., McCallum, A.: An Introduction to Conditional Random Fields for Relational Learning (2006)