Toolset for image and text processing and metadata editing – Initial release

Petr Sojka, Radim Hatlapatka, Josef Baker, Łukasz Bolikowski, Thierry Bouche, Nicolas Houillon, Michael Jost, Michal Ružicka, Aleksander Nowiński, Alan P. Sexton, et al.

EuDML
The EUROPEAN DIGITAL MATHEMATICS LIBRARY

ICTPSP
ICT POLICY SUPPORT PROGRAMME
part of the Competitiveness and Innovation Framework Programme CIP

DEMO

Project Acronym: EuDML

Grant Agreement number: 250503

Project Title: The European Digital Mathematics Library

D7.2: Toolset for Image and Text Processing and Metadata Editing – Initial release

Revision: 1.0 as of 1st March 2011

Authors:

Petr Sojka	Masaryk University, MU
Radim Hatlapatka	Masaryk University, MU

Contributors:

Josef Baker	University of Birmingham, UB
Łukasz Bolikowski	ICM Warsaw, ICM
Thierry Bouche	UJF/CMD
Nicolas Houillon	UJF/CMD
Michael Jost	FIZ Karlsruhe, FIZ
Aleksander Nowiński	ICM Warsaw, ICM
Michal Růžička	Masaryk University, MU
Alan Sexton	University of Birmingham, UB
Volker Sorge	University of Birmingham, UB

Project co-funded by the European Comission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	✓
C	Confidential, only for members of the consortium and the Commission Services	

Revision History

Revision	Date	Author	Organisation	Description
0.1	24 November 2010	Petr Sojka	MU	First version with structure of the deliverable.
0.2	27th January 2011	RH,PS	MU	MU tools' sections added, structure revised.
0.3	16th January 2011	PS,JB,MJ	MU,UB,FIZ	Contribs by UB, FIZ added, structure revised.
0.4	15th January 2011	PS	MU	Cleanup, actual version.
0.5	17th February 2011	TB/NH	UJF/CMD	Typos corrected, CMD tools added
0.6	24th February 2011	PS,MR,LB	MU,ICM	FIZ update, ICM part and summary added, further corrections.
0.7	25th February 2011	PS,MR	MU	Interim version close to internal review, TODOs fixed.
0.8	25th February 2011	PS	MU	Version for internal review.
0.8+	25th February 2011	PS	MU	Some references added, contributors sorted.
0.9	27th February 2011	AS	UB	Corrected many small errors.
0.9+	1st March 2011	PS,RH,MR,LB	MU	Version with comments reflected.
1.0	1st March 2011	Petr Sojka	MU	Release for EU.

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Contents

Executive Summary

This demonstration description presents tools produced by EuDML partners and made available for demonstration. They demonstrate building bricks of enhancer tools, whose functionality should check, correct and enhance metadata collected both from partners, including Zentralblatt MATH, and from the analysis of full text or PDF document versions of items in the EuDML collection. Demonstration web pages allow testing and evaluation of thirteen tool prototypes.

`http://nlp.fi.muni.cz/projekty/eudml/eudmldemo.php`

`http://demo.eudml.eu/demo/`

`http://www.zentralblatt-math.org/EuDML-test1?`

`http://thar.ujf-grenoble.fr/EuDML/demo/NLMTeX2TeX+MML/`

`http://thar.ujf-grenoble.fr/EuDML/demo/TeX2NLM/`

`http://www.cs.bham.ac.uk/research/groupings/reasoning/sdag/eudml-demo.php`

`http://wysoka.icm.edu.pl:18190/EuDmlAnalysisDemo/`

"All great masters are chiefly distinguished by the power of adding a second, a third, and perhaps a fourth step in a continuous line. Many a man had taken the first step. With every additional step you *enhance* immensely the value of your first."

Ralph Waldo Emerson (American Poet, Lecturer and Essayist, 1803–1882)

1 Introduction

This demonstration deliverable describes the initial version of a toolset for image and metadata enhancement and editing. The tools needed were identified in D7.1 [28] and their basic versions were developed and tested mostly at technology provider's sites.

The purpose of this demonstration is to verify and discuss functionality, usability, scalability, effectiveness and effectivity of every tool towards building a comprehensive workflow based on a series of data enhancers.

The handling and tools to cope with the [meta]data are quite different, depending on primary data origin. Figure 1 shows the top-level enhancement tool structure, with *subsystems* represented as edges.

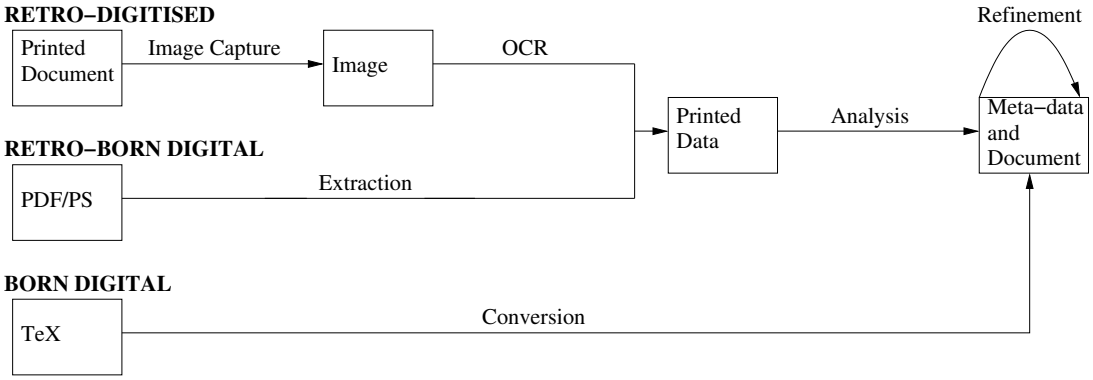

Figure 1: Metadata processing

1.1 The Toolset

The toolset consists of five subsystems: OCR, Extraction, Analysis, Conversion and Refinement, and a set of external tools offered to data providers (External). Subsystems are built based on the smaller bricks of software — *eutools* — as defined in [28, Section 1.3]. At this stage, eutools are being developed and tested mostly at the technology providers' sites, with well defined interfaces allowing further integration into subsystems on the EuDML core system site. The toolset currently consists of the eutools listed in Table 1 on the facing page.

Table 1: Eutools version 0.1 overview

Subsystem	Partner	Eutool	Functionality
OCR	MU	**PdfToTextViaOCR**	Basic plaintext extraction from bitmap images which are rendered from a PDF document
Extraction	UB	**PDFTester**	Tests whether a PDF document contains multiple layers, page bitmaps and/or is born-digital
Extraction	ICM	**PDF Text Extractor**	Extracts plain text from a PDF document
Extraction	UB	**MathML Extractor**	Analyses a PDF document and extracts the mathematical expressions from it as a list of MathML structures
Conversion	CMD	**NLMTeX2TeX+MML**	Takes an NLMTeX file and adds MathML translations for all mathematical expressions encoded as LaTeX sources in the file.
Conversion	CMD	**TeX2NLM**	Identifies and tags TeX math in a CDATA UTF-8 string
Analysis	ICM	**Plain Text Reference Segmenter**	Extracts bibliographic references from plain text.
Analysis	ICM	**Bibliographic Reference Parser**	Parses a plain text bibliographic reference (extracts author names, title, publication year, etc.)
Refinement	FIZ	**ZBMATH Metadata Lookup**	Get Zentralblatt MATH metadata for a publication given its Zentralblatt identifier
Refinement	MU	**PdfJbIm**	Recompress bitmap streams in a PDF document with JBIG2
Refinement	MU	**Pdfsizeopt**	Optimize PDF documents
Refinement	MU	**math_metadata_lookup**	MR/Zbl metadata search and fetch
External	MU	**ME**	Metadata Editor — standalone editing for use at providers' sites

1.2 Structure of the Demo

The very important issue is the interface the eutools will communicate within WP7 with other eutools and with the EuDML core system. In the next section we do list possibilities of possible and preferred interfaces to be used within EuDML. Section 3 lists 13 eutools that have been prototypically implemented, together with the web pages that demonstrate their functionality. Section 4 sums up demoed tools, achievements and reviews future direction of development of EuDML enhancers and issues to be tackled soon.

2 API Definitions

An API (Application Programming Interface) is an interface that is used by programs to communicate with each other. There are several types of interfaces that can be used in EuDML.

We can distinguish tool interfaces by the number of documents they are applied to:

- Batch processing
  - This is used for running on a large set of documents with the same parameters.
  - Parameters are set in configuration files or statically in a script or program.
  - This class includes indexers, where parameters for indexing are set in a configuration XML file.
- Individual (manual) processing
  - Parameters are set for each document separately.
  - These tools require a significant level of interaction.
  - This class includes tools using OCR (e.g. **PdfToTextViaOCR** see 3.1), where it may be required to set the language for each of the documents.
  - Should it be possible to get required parameters dynamically, for example via some other tool, then the tool may be used for batch processing as well.

Interface types can be based on how the main document or data type for processing are interchanged (e.g. PDFs, XML):

- Input/output streams (e.g. **PdfJbIm**)
- Files (used by, e.g. **PdfJbIm**, **Pdfsizeopt**, **NLMTeX2TeX+MML**)
- Java objects (e.g. **PDFBox**)
- Remote procedure calling (RPC), encoding parameters using XML. For example: RMI (Remote Method Invocation) or SOAP.
- Using command line (e.g. **MathML Extractor** see 3.4)

For correct usage some additional parameters are often required which modify the behavior of a tool. They can be given in one of a number of forms:

- A configuration file — the parameters are set in a specific file (usually XML). This approach is useful mostly for batch processing.
- A set of arguments — put directly as additional parameters of a called method.
- An array of parameters — a keyword specifying a name of a parameter plus a value to be set (e.g. **Pdfsizeopt**).

Within the EuDML project a number of different platforms (programming languages) are used whose interfaces can differ significantly. The most relevant platforms are:

- Java — using Java objects and Java interfaces.
- C/C++ (Tralics).
- Python — Jython (Java interpreter for Python) can be used from Java. (e.g. **Pdfsizeopt**)
- Command line (Bash, Perl)
- Lisp or OCaml (e.g. **MathML Extractor**)

The most common formats for interchanging parameters between platforms are:

- JSON (JavaScript Object Notation)
- XML (e.g. NLM)

Because the core of the EuDML project is being developed in Java, it is preferred to use Java platform and interfaces. As many eutools and software libraries were already developed in other languages (OCaml, C, C++, Python,...) and it would be expensive to rewrite them in Java. That's why it is suggested for bigger tools, where the bottleneck of efficiency is not their invocation using Java system call, to encapsulate them and modify them for this kind of usage.

If this is too expensive (from efficiency point of view) there is possibility to use JNI (Java Native Interface). If the bottleneck is not communication through network RMI could be used as well.

3 Eutools

In this section basic information is provided for every tool, including their defined input and output interfaces, license information, programming language and evaluation, as well as the URL of the demonstration web site.

Several tools described in this section are so-called *processing nodes*, which can be chained together into so-called *processes*. The initial node in a process typically generates or otherwise obtains chunks of data which are consecutively processed by the following nodes. A node typically enhances the chunks that it receives on its input and sends the enhanced chunks to its output, possibly with side effects such as indexing the contents of the chunk. The final node in a process typically stores the enhanced chunk in a storage or discards it. There is a processing framework written in Java which orchestrates the flow of data chunks between nodes. Therefore, an author of an individual tool only needs to implement a processing node with well-defined inputs and outputs.

3.1 PdfToTextViaOCR

PdfToTextViaOCR is a tool written in Java which extracts images from a PDF document and uses an OCR engine to render text from them. The currently employed OCR engine is Tesseract [20]. This tool is licensed under Apache License 2.0 `http://www.apache.org/licenses/LICENSE-2.0`.

The process of extracting text from PDF documents using OCR consists of two steps:

1. Extract images from a PDF document.
2. Run Tesseract with appropriate parameters on the resulting images.

In the future we intend to insert an image preprocessing step to improve quality of extracted images and prepare them for OCR analysis by Tesseract.

Some additional parameters, such as the language of the text, should be set to achieve better output text quality. Since version 3.0, Tesseract supports a very large number of languages, some of which use a Cyrillic alphabet. If there is a language which is not supported yet, it is possible to train Tesseract and thus enhance its language support.

The main benefit of using Tesseract is its wide language support and that it is an open-source OCR with one of the highest accuracies. The disadvantage is lack of supporting OCR of math. In the future the intention is to use an OCR engine capable of handling mathematics as well.

PdfToTextViaOCR inputs a PDF document either as a file or as an input stream and renders text from the bitmap images stored in the PDF. The rendered text is then returned in set of files, one text file per image in PDF (usually each image represents a separate page).

PdfToTextViaOCR shall be used to extract text to be used mainly for indexing and searching inside EuDML, and to provide text to other WP7 or WP10 tools (Braille drivers etc.).

A demonstration version of this tool is available at `http://nlp.fi.muni.cz/projekty/eudml/eudmldemo.php`.

3.2 PdfTester

PDFTester is a tool written in OCaml that determines how a given PDF file should be processed. The software requires an uncompressed PDF file which is currently created via a call to PDFTK, which is licensed under the GNU General Public License Version 2. There are three possible outcomes of running the tester, namely;

- The PDF contains multiple layers which should be extracted automatically.
- The PDF contains sufficient information to automatically extract contents and generate MathML.
- The PDF should be processed via rendering to an image and the use of OCR.

The first check to identify multiple layers within the PDF is completed by reading the content streams within the file and searching for marked content. If marked content is found then the tool will return an integer indicating that the file has multiple layers.

The second check to identify the compatibility of the file with the **PDF Text Extractor** tool is completed by extracting the fonts and content streams from the file. If these can be parsed successfully, then an integer is returned indicating that the file is compatible.

If both of these checks fail, then the file is unsuitable for enhanced analysis by the current EuDML tool set. Therefore an integer is returned indicating it should be rendered to an image and processed via the **PdfToTextViaOCR** tool.

3.3 PDF Text Extractor

PDF text extractor is written in Java, and uses the Apache PDFBox library [25] to obtain plain text from a PDF document. PDFBox is an open source software distributed under Apache License Version 2.0.

It is implemented as a *processing node*, which effectively takes a document identifier on input and produces plain text on output.

The processing node works as follows. First, it fetches the corresponding PDF file from the content storage, or from a local cache, if present. Next, it uses `PDFTextStripper` from Apache PDFBox to extract plain text. Finally, the plain text is scheduled for addition to the content storage under a relevant identifier.

Text extraction from PDF files is a prerequisite for other tasks in the EuDML, including:

- extraction of bibliographic references from plain text;
- full-text indexing of the entire collection.

A joint demonstration of text extraction from PDF documents, bibliographic reference extraction from plain text, and bibliographic reference parsing is available at `http://wysoka.icm.edu.pl:18190/EuDmlAnalysisDemo/`.

3.4 MathML Extractor

MathML Extraction from PDF documents are handled by a tool written in OCaml that returns the mathematical expressions from a PDF as a list of MathML structures. The software requires an uncompressed PDF file which is currently created via a call to PDFTK.

Given an appropriate PDF file, of which the suitability can be determined by running **PDFTester**, the expressions on either a single page or the whole document can be extracted. The tool works by extracting the fonts and content streams from a PDF which are then parsed by the **MathML Extractor**, producing a list of symbols and graphics for each page. These, in conjunction with a list of glyphs obtained via image analysis of the page images rendered from the PDF document, are used to split each page into a number of lines. Each line is parsed to create a parse tree, then processed by a driver that separates text from in-line math expressions and produces MathML markup for any formulae that occur on that line. The software is based upon the work described in [6, 5, 7]. However, it features automatic segmentation, based upon spacing and font information, rather than requiring manual clipping of formulae as stated in those papers.

A limitation of the tool is that it can only work with PDF files making use of Type 1 fonts and embedded font encodings. This generally means that the file will have been generated from LaTeX, Troff or a number of other document production systems, which does limit the number of potential sources. Also, the segmentation process is still in development and may miss small inline formulae and will also split and not align larger, multiline formulae. This will be addressed in subsequent versions of the tool.

The URL of the demonstration web site is `http://www.cs.bham.ac.uk/research/groupings/reasoning/sdag/eudml-demo.php`.

In future versions of the toolset, extraction eutools will be merged into one, and some eutools will become superfluous.

3.5 NLMTeX2TeX+MML

Most project partners are not able to provide metadata in NLM directly, including MathML for formulae. Based on the experience from CEDRICS [9] and DML-CZ [27],

two tools were designed and prototypes thereof were implemented using Tralics: **NLMTeX2TeX+MML**, described in this section, and **TeX2NLM**, described in Section 3.6. The former tool adds math formulae in MathML from TEX strings in NLMTeX documents, the latter identifies and marks such TEX strings in a UTF-8 string.

NLMTeX2TeX+MML takes, as input, a valid XML file where mathematical formulas are enclosed in an <inline-formula> or <disp-formula> element with its internal EuDML v1.0 DTD structure [21], containing a TEX-encoded version of the formula in the <tex-math> element. It returns the same file with unchanged content except that a <mml:math> element is added as an alternative within <*-formula>, with a MathML representation of the formula derived from the provided TEX version.

It is meant as a batch tool that will upgrade any existing metadata with (presentation) MathML for any properly tagged formula written in TEX.

The tool's interface is written in PHP and Java and awaits a definitive API to be integrated in EuDML system through REST or SOAP calls.

At its core, **NLMTeX2TeX+MML** relies on Tralics [15], which is a C++ program that reads a text file and writes an XML file. This needs to be configured in order to handle the standard mathematical constructs that are supported by the tool (macro namespace of plainTEX, base LATEX with AMS-LATEX extensions): a number of supporting macros and configuration files have to be known to the Tralics compiler, which are stored in a small disk space that Tralics must be able to access (paths can be given in the command line). Of course, if the tool is run on a dedicated server, these details can be hidden from the rest of the system.

Tralics is free software governed by the CeCILL license that can be found at `http://www.cecill.info/`. Tralics version 2.13.6 or higher is required, the more recent version 2.14.1 is under evaluation.

The assumptions for this tool are the following:

- It is applied to a well-formed XML file (it can be a valid XML file conforming to EuDML specification v1.0, but could also be just a fragment thereof).
- In this XML file, all formulae are tagged with the NLM JATS basic structure:
  - a mandatory <inline-formula> (or <disp-formula> for displayed math), with a mandatory *@id* attribute (unique identifier);
  - an optional <alternatives> element if the formula has variant encodings;
  - a mandatory <tex-math> element holding valid TEX code (with & and < escaped using & and < entities);
  - The TEX commands switching to math mode must be explicit in the TEX code as $ in the supplied example (it could also be `\[..\]`, `\begin{align}`, etc.);
  - Once unescaped, the TEX code contains no unspecified macros and compiles with allowed and configured TEX commands.

Example 1: An input formula with TEX-only encoding in NLM structure

```
<inline-formula id="d18e3147">
  <tex-math>$J_k(n) := n^k \prod _{p \mid n}(1 - p^{-k})$</tex-math>
</inline-formula>
```

Example 2: An input formula with TEX encoding and alternative image in NLM structure

```
<inline-formula id="d18e3148">
  <alternatives>
    <tex-math>$J_k(n) := n^k \prod _{p \mid n}(1 - p^{-k})$</tex-math>
    <graphic xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="fors2682.f3">
      <object-id>463492</object-id>
    </graphic>
  </alternatives>
</inline-formula>
```

The tool's output for the first example

```
<inline-formula id="d18e3147">
  <alternatives>
    <mml:math xmlns="http://www.w3.org/1998/Math/MathML">
      <mml:mrow>
        <mml:msub>
          <mml:mi>J</mml:mi>
          <mml:mi>k</mml:mi>
        </mml:msub>
        <mml:mrow>
          <mml:mo>(</mml:mo>
          <mml:mi>n</mml:mi>
          <mml:mo>)</mml:mo>
        </mml:mrow>
        <mml:mo>:</mml:mo>
        <mml:mo>=</mml:mo>
        <mml:msup>
          <mml:mi>n</mml:mi>
          <mml:mi>k</mml:mi>
        </mml:msup>
        <mml:msub>
          <mml:mo>?</mml:mo>
          <mml:mrow>
            <mml:mi>p</mml:mi>
            <mml:mo>&mid;</mml:mo>
            <mml:mi>n</mml:mi>
          </mml:mrow>
        </mml:msub>
        <mml:mrow>
          <mml:mo>(</mml:mo>
          <mml:mn>1</mml:mn>
          <mml:mo>-</mml:mo>
          <mml:msup>
            <mml:mi>p</mml:mi>
            <mml:mrow>
              <mml:mo>-</mml:mo>
              <mml:mi>k</mml:mi>
            </mml:mrow>
          </mml:msup>
          <mml:mo>)</mml:mo>
        </mml:mrow>
      </mml:mrow>
    </mml:math>
    <tex-math>$J_k(n) := n^k \prod _{p \mid n}(1 - p^{-k})$</tex-math>
```

```
  </alternatives>
</inline-formula>
```

An obvious limitation of this tool is that it assumes formulae to be clearly identified and tagged in the input files. Apart from *generated* TEX formulas, it is not likely that such structure will be present in existing metadata, and it is not easy nor reliable to produce such structure from generic TEX code (mixing text and math modes) without a fully-fledged TEX engine at hand.

An obvious advantage of the well-defined input format is that the tool is not likely to fail as it only has to support the subset of the TEX (and AMS-LATEX) syntax dealing with mathematical formulas, which is much more constrained and predictible than TEX as a document processing system.

The demonstration version of **NLMTeX2TeX+MML** is available at `http://thar.ujf-grenoble.fr/EuDML/demo/NLMTeX2TeX+MML/`.

3.6 TeX2NLM

TeX2NLM takes, as input, a UTF-8 encoded CDATA string that is expected to be valid TEX code and returns the same content with formulas identified as such and conforming to EuDML NLM structure (a <*-formula> element for each formula, containing an <alternatives> element with a child <tex-math> element holding the original TEX code, <mml:math> element with a MathML representation of the formula derived from the provided TEX version.

This tool is meant for the (quite typical) case when a content provider has not identified formulas included in textual metadata fields such as titles, abstracts or keywords, and thus cannot make use of the **NLMTeX2TeX+MML** tool. The intended use of this tool is to process some fields that are known to contain unidentified TEX formulas so that the NLM representation can be made more usable. It is not yet obvious whether it can be integrated in some component of the EuDML enhancing workflow, or if it would be better used as an external preprocessor before the providers' metadata is converted to NLM and contributed to the EuDML system.

The tool's interface is written in PHP and Java and awaits a definitive API to be integrated into the EuDML system through REST or SOAP calls. As the previous tool **TeX2NLM** relies on Tralics [15].

The assumptions for this tool are the following:

- It is applied to a flat CDATA string (typically, the content of a childless XML element holding textual information with TEX-encoded mathematical formulas).
- This string must be valid and self-contained TEX code, text being UTF-8 encoded (with & and < escaped using & and < entities).

The tool identifies each included formula and generates a standard NLM structure for each of them.

Example: An input string with a TEX formula

```
The formula $J_k(n) := n^k \prod _{p \mid n}(1 - p^{-k})$ clearly defines nothing.
```

The tool's output for this example

The formula

```
<inline-formula id="d18e3149">
  <alternatives>
    <mml:math xmlns="http://www.w3.org/1998/Math/MathML">
      <mml:mrow>
        <mml:msub>
          <mml:mi>J</mml:mi>
          <mml:mi>k</mml:mi>
        </mml:msub>
        <mml:mrow>
          <mml:mo>(</mml:mo>
          <mml:mi>n</mml:mi>
          <mml:mo>)</mml:mo>
        </mml:mrow>
        <mml:mo>:</mml:mo>
        <mml:mo>=</mml:mo>
        <mml:msup>
          <mml:mi>n</mml:mi>
          <mml:mi>k</mml:mi>
         </mml:msup>
         <mml:msub>
           <mml:mo>?</mml:mo>
           <mml:mrow>
             <mml:mi>p</mml:mi>
             <mml:mo>&mid;</mml:mo>
             <mml:mi>n</mml:mi>
           </mml:mrow>
         </mml:msub>
         <mml:mrow>
           <mml:mo>(</mml:mo>
           <mml:mn>1</mml:mn>
           <mml:mo>-</mml:mo>
           <mml:msup>
             <mml:mi>p</mml:mi>
             <mml:mrow>
               <mml:mo>-</mml:mo>
               <mml:mi>k</mml:mi>
             </mml:mrow>
           </mml:msup>
           <mml:mo>)</mml:mo>
         </mml:mrow>
      </mml:mrow>
    </mml:math>
    <tex-math>$J_k(n) := n^k \prod _{p \mid n}(1 - p^{-k})$</tex-math>
  </alternatives>
</inline-formula>
```

clearly defines nothing.

As Tralics is a fully-fledged TEX engine, it should be able to handle reasonably many existing TEX metadata to convert them to NLM with both <tex-math> and <mml:math>. The big advantage of Tralics is that it will reliably identify all possible TEX math mode switching commands, thus generating clean and correct NLM structures for each formula encountered. The drawback is that the TEX code encountered must be very clean and

standard, as Tralics will compile it to generate the alternatives. Unknown macros, which are quite frequent when the source TEX code was produced by the original author, rather than a publisher, will make the string unprocessable at all.

Another lightweight approach would be to use heuristics to detect formulae in the input string, generate the NLM structure and then pass the resulting XML to **NLM-TeX2TeX+MML** in order to get MathML. The drawback with this method is that heuristics are bound to be fooled by the full power of the TEX language. However, it might well be that this could succeed on the actual data to be processed. We will compare and assess the relative efficacy of both methods.

A demonstration version of **TeX2NLM** is available at `http://thar.ujf-grenoble.fr/EuDML/demo/TeX2NLM/`.

3.7 Plain Text Reference Segmenter

Some document metadata do not contain bibliographic references, yet it is valuable information that can be used, among others, for improved navigation in user interface, easy linking and, eventually, for further bibliometric analysis.

Plain Text Reference Segmenter extracts bibliographic references from plain text. From the system integration point of view, it is implemented as a processing node that takes, as input, the plain text of a document (from a cache or from the storage) and an NLM metadata record. The segmenter enhances the NLM record with bibliographic references by adding `<mixed-citation/>` tags and outputs the enhanced NLM record. The tool is written in Java and does not depend on any third-party libraries.

This implementation of bibliographic extraction is based on the observation that certain characters, such as: comma, dot, colon, or parentheses occur frequently in the bibliography, so abundance of such characters in a line is a hint that the line may be in the references section of an article. Therefore, the algorithm begins with "marking" the lines that might reside in the bibliographic references section of the document. Next, the region of the text with the largest concentration of such lines is assumed to be the references section. Finally, the section is split into individual references. To meet this end, several patterns are tested and line lengths are examined in order to deduce where one reference ends and another one begins. The current implementation is a proof-of-concept prototype that needs additional tuning and evaluation. The tool is demonstrated at `http://wysoka.icm.edu.pl:18190/EuDmlAnalysisDemo/`.

3.8 Bibliographic Reference Parser

Metadata sources often provide bibliographic references in the form of raw, untagged text. In order to navigate the references in a user interface or to analyze the citation network, it is necessary to parse the raw texts of references into fragments such as: author, title, journal, volume, year, etc. For example, the following input text:

```
Š. Višňovský, Czech. J. Phys. B 36, 625 (1986)
```

should be parsed as follows (here in the RIS exchange format):

```
TY - JOUR
AU - Višňovský, Š.
JO - Czech. J. Phys. B
```

```
VL - 36
SP - 625
PY - 1986
```

However, reference parsing is not a trivial task, for several reasons:

- There are dozens of established reference formats, and a great variety of formats "invented" by authors.
- Reference texts are "noisy" due to: misspellings, OCR errors and imperfect transformations from one format to another (the latter especially affects characters with diacritical marks and mathematical formulas).
- Interpretation of a reference is sensitive to punctuation: a single comma changed to a colon may alter the meaning of a whole citation.

A number of approaches to reference parsing have been developed. Template matching using regular expressions is arguably one of the earliest techniques. It is actively used to this day, for example in the ParaCite project [1]. The BibPro tool [10] uses sequence alignment algorithm BLAST [3] to find the best-fitting reference template. Several machine learning approaches exist, based on Hidden Markov Models [19], Conditional Random Fields [30], and other probabilistic models. Last but not least, there are efforts to combine the above techniques [16].

In EuDML, the current implementation of reference parser is based on regular expressions (a future version may be based on Conditional Random Fields). It is implemented in Java, as a processing node which takes, as input, an NLM metadata record and returns a similar NLM record in which all the unparsed references are now parsed. The tool does not use any third-party libraries.

A joint demonstration of text extraction from PDF, bibliographic reference extraction from plain text, and bibliographic reference parsing is available at `http://wysoka.icm.edu.pl:18190/EuDmlAnalysisDemo/`.

3.9 ZBMATH Metadata Lookup

ZBMATH Metadata Lookup is a special online interface to the Zentralblatt MATH (ZBMATH) database [12] which, given a Zentralblatt MATH item identification number (ZblID), returns the metadata that are stored in the ZBMATH database for this publication.

Access to this interface is free for EuDML project partners. EuDML project partners may use metadata from ZBMATH to enhance or refine metadata for publications provided by them.

The tool provides a HTTP query interface that takes as input the ZblID of the item in question (URL encoded, see example below). The interface returns XML-encoded ZBMATH metadata for that item, MathML encoded where possible, and UTF-8 otherwise (i.e., no TEX is used). The question of whether the data exposed should be converted to the EuDML metadata format based on NLM JATS DTD [21], or better delivered in a structure that closely reflects the ZBMATH internal data format has been discussed, and it was resolved that during the current stage of the project the latter option is more suitable. A processing node that fetches a Zentralblatt record whenever a ZblID is encountered in the EuDML metadata was written by ICM, as part of the EuDML workflow. ICM

has also written a processing node that calls a Zbl-to-EuDML metadata converter (to be implemented by IST) and stores the resulting EuDML metadata for further processing.

The tool is written in Python as part of the Zentralblatt MATH proprietary search engine. Access is possible via a HTTP interface. For example, `http://www.zentralblatt-math.org/EuDML-test1?query=an:1163.57016` returns the following piece:

```
<result query="an:1163.57016" hits="1">
  <item no="1">
    <ZblId>05549525</ZblId>
    <ZblNo>1163.57016</ZblNo>
    <Origin>Zbl</Origin>
    <DocType>serial_article</DocType>
    <PublicationYear>2009</PublicationYear>

    <Links>
      <Link>doi:10.1007/s00229-008-0246-z</Link>
    </Links>
    <Authors>
      <Author>Wegner, Christian</Author>
    </Authors>
    <Title><span><math
        xmlns='http://www.w3.org/1998/Math/MathML'><msup><mi>L</mi> <mn>2</mn>
        </msup></math></span>-invariants of finite aspherical CW-complexes.</Title>

    <Languages>
      <Language>EN</Language>
    </Languages>
    <MSCCodes>
      <MSCCode>57Q10</MSCCode>
      <MSCCode>55N99</MSCCode>
    </MSCCodes>

    <Keywords>
      <Keyword><span><math
          xmlns='http://www.w3.org/1998/Math/MathML'><msup><mi>L</mi>
          <mn>2</mn> </msup></math></span>-torsion</Keyword>
      <Keyword>finite classifying spaces of amenable groups</Keyword>
      <Keyword>localization</Keyword>
      <Keyword>Novikov-Shubin invariants</Keyword>

    </Keywords>
    <Source>Manuscr. Math. 128, No. 4, 469-481 (2009).</Source>
    <SerialTitle>Manuscripta Mathematica</SerialTitle>
    <SerialShortTitle>Manuscr. Math.</SerialShortTitle>
    <VolumeNumber>128</VolumeNumber>
    <IssueNumber>4</IssueNumber>

    <Pages>469-481</Pages>
    <Publisher>Springer-Verlag, Berlin</Publisher>
    <ISSN>0025-2611; 1432-1785</ISSN>
  </item>
</result>
```

3.10 PdfJbIm

PdfJbIm is a PDF enhancer written in Java which reduces the size of PDF documents containing bitonal images [18]. It takes advantage of the extremely high compression ratio of visually lossless JBIG2 compression [11].

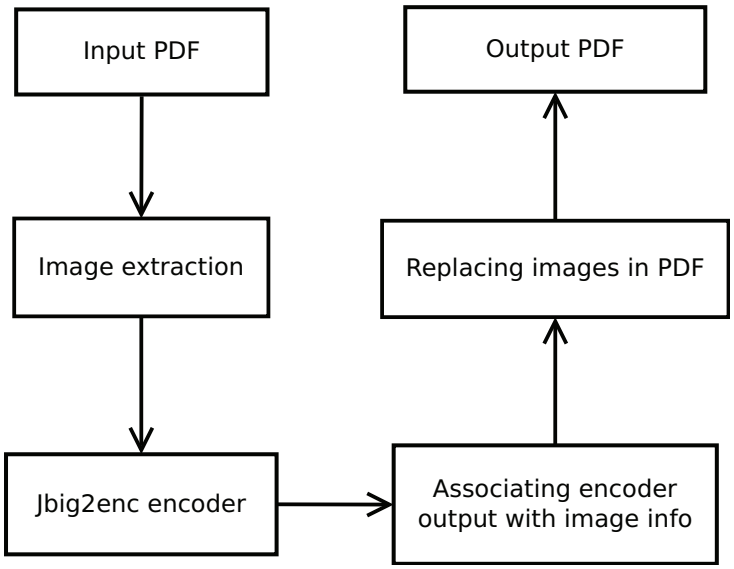

Figure 2: Workflow of **PdfJbIm**

PdfJbIm uses the external open-source encoder jbig2enc [23], developed by Adam Langley. The compression ratio of jbig2enc has been improved by about ten percent by creating an additional comparison process for distinguishing representatives of symbols. For more information about **PdfJbIm** and the modification of jbig2enc see [17, 29]

The PDF document input to **PdfJbIm** can be given either as a Java input stream or as a file. A path to the jbig2enc encoder executable must be set, which is used in the process of optimizing the PDF. It is possible to modify the behaviour of **PdfJbIm** by setting additional parameters. For example it is possible to specify whether or not non-bitonal images should be re-compressed as well. All images re-compressed by **PdfJbIm** are stored as bitonal in the output PDF. Output of **PdfJbIm** is a PDF file with reduced size but without visible loss of data. The output PDF is returned either as file or as an output stream.

On average it reduces the size of PDF originally compressed using Fax G4 by thirty percent. This statistic was found by running **PdfJbIm** on the journals stored at DML-CZ. The size of the journals was reduced from 20,593 MB to 14,653 MB. This result was presented also in [29]. **Pdfsizeopt** can be included in the workflow containing **PdfJbIm**, thus reducing PDF document sizes even further. For more information see Section 3.11.

This tool can be used by content providers or in the EuDML core. In both cases it significantly reduces the storage size and, therefore, the time required to access and download a PDF document from a remote repository.

A demonstration version of **PdfJbIm** is available at `http://nlp.fi.muni.cz/projekty/eudml/eudmldemo.php`.

3.11 Pdfsizeopt

Pdfsizeopt [26] is not a tool in the classic meaning of the word. It is a collection of best practices and Unix scripts to optimize the size of PDF documents. It is being developed in Python by Peter Szabó with support of Google.

For optimization of each type of object in PDF, several methods are attempted and the one giving the best result is chosen. The disadvantage of this approach is longer runtime but it achieves better results.

Specific Unix tools are used for optimizing different parts of PDF documents. What can be optimized and how is based on which tools are installed. Most tools which are used by **Pdfsizeopt** can be disabled. Lots of PDF optimizing is done by Multivalent, which is therefore a primary tool that should be installed and enabled for use in **Pdfsizeopt**. On the other hand, it is better to disable `pngout`, because the improvement gained by using it is minimal compared to runtime.

Pdfsizeopt uses jbig2enc for optimizing bitonal images but uses generic coding even for a text region. Thus it does not achieve as high a compression ratio as if symbol coding were being used. Therefore it is better to disable it and use **PdfJbIm** instead, which uses symbol coding for text regions in the image and therefore achieves better compression ratio for images containing scanned text. For more information see [29].

Python must be installed to run **Pdfsizeopt**. It takes a PDF file as input and creates a new PDF file. It takes additional parameters which enables the specification of which tools should be used. These parameters are given in the format `--use-${tool name}=${boolean value}` where `${tool name}` is the name of the tool in question and `${boolean value}` is true or false to indicate whether the tool should be used. Default values are used if the parameter is not explicitly set for a tool.

For better integration with Java, thus simplifying usage by EuDML core system, it is possible to run scripts written in Python using Jython which is an interpreter of Python running under the JVM (Java Virtual Machine).

3.12 math_metadata_lookup

The **math_metadata_lookup** program [22] is a small open-source (GNU LGPL [14]) command-line utility implemented in Ruby. It searches through mathematical reviews databases and fetches metadata. Supported reviews databases are Mathematical Reviews [4] (MR licensed access is necessary) and Zentralblatt MATH [12] (full metadata set is available *only if* Zbl licensed access is used, otherwise results are suboptimal) but it should be easy to add another database when necessary.

The **math_metadata_lookup** utility provides their users with four different search options:

1. article search — Search and fetch article metadata from reviews databases according to title, authors, year, or ID parameters given by the user.
2. author search — Search reviews databases and fetch all the different name forms for given author name. The database author ID and preferred name form is also given.
3. heuristic search — This is similar to the article search but only one (the best) match from each database is returned. Moreover, similarity of given parameters (title, authors, year) and found records must be higher than a threshold given by the user.

Parameters given by the user are not used directly but are preprocessed to increase the probability of a search hit. Similarity of found records and original user given search parameters is computed using a generalized Levenshtein edit distance [24].

4. reference search — In this mode, the **math_metadata_lookup** utility tries to parse a user given reference string and identify title, author and year fields. These parameters are then used for a heuristic search. A user specified threshold is used to filter out records with insufficient similarity to the original search parameters.

Records found by the **math_metadata_lookup** utility are printed to the standard output in the desired format: plain text, HTML, XML, Ruby, or YAML.

A demonstration version of the tool is available at `http://nlp.fi.muni.cz/projekty/eudml/eudmldemo.php`.

3.13 Metadata Editor

The Metadata Editor, **ME**, [2, 8] is an open-source (GNU GPL [13]) client–server web application implemented mainly in Ruby and Perl programming languages. It is designed to manage, edit and validate metadata and full texts of digital publications prior to their integration into a digital library.

The main input of **ME** is a collection of scanned pages of digitized publications. Once imported to **ME** the pages can be distributed among articles that themselves are assigned to issues, volumes and journals. On each level of the structure, **ME** provides their operators with visual presentation of the content and allows the operators to fill in description (metadata). An integrated search tool allows users to search through the Metadata Editor metadata repository. Automatic tests can be used to verify properties and completeness of the metadata. An integrated authority database allows unambiguous assignment of publications to particular people. **ME** can be used remotely and simultaneously by several operators. Access to **ME** functions and repository content can be restricted to particular users using an access rights system.

ME is not intended to be a part of the EuDML Core — it is a stand-alone application that enables EuDML data providers without their own solution to organize and annotate their digitized publications and prepare their data to reach at least the entry level (*basic metadata* in WP3 jargon) for the EuDML project.

A demonstration version of this tool, together with links to download it is available at `http://nlp.fi.muni.cz/projekty/eudml/eudmldemo.php` where it can be tried out (login/password: `admin`/`admin`).

4 Summary, Conclusions

A summary of the eutools prepared is shown in Tables 2 and 3. Eutools now will be merged into larger components (e.g. one PDF reading tool) and tested on the central EuDML site on real data from providers.

Most enhancers will be used internally in the main EuDML architecture, with the exception of **ME**, which will be offered to potential partners. Enhanced PDFs might be offered to partners via agreed interfaces. Otherwise they will be used for EuDML internal purposes, mainly by WP7–WP10 toolsets.

Table 2: WP7 Eutools integration summary – OCR, Extraction, Conversion, and Analysis

Tool name	Input	Output	Main benefit for EuDML
		OCR subsystem	
PdfToTextViaOCR	PDF with images as input stream or file (content/<file name>)	Set of plaintext/text files	OCR-ed text suitable for indexing
		Extraction subsystem	
PDFTester	Any PDF file	Integer recording whether the file (a) has an additional layer for extraction, (b) can be used with the MathML extractor or (c) should be processed via OCR	Indicates most suitable tool for PDF analysis
PDFBox	Article PDF	Article fulltext	Extracts text from born-digital PDF (without using OCR)
MathML Extractor	Suitable PDF, as indicated by **PDFTester**	List of MathML fragments for each formula within the file	Indexable, accessible MathML from a standard PDF
		Conversion	
NLMTeX2TeX+MML	XML file	XML with MathML for all tagged formulas	Upgrades TEX encoding of NLM-tagged formulas to MathML
TeX2NLM	UTF-8 encoded CDATA string	UTF-8 encoded PCDATA with TEX formulas converted to NLM structure with TEX and MathML alternatives	Upgrades untagged TEX formulas to full TEX+MathML NLM structure
		Analysis subsystem	
Plain Text Reference Segmenter	article plain text part with references	segmented set of references	Provides bibliographic references from plain text
Bibliographic Reference Parser	CDATA string of a reference	parsed reference in NLM XML	Identifies author names, title, publication year in a reference string

Table 3: WP7 Eutools integration summary – Refinement and External

Refinement subsystem			
ZBMATH Metadata Lookup	Zbl identifier	Metadata in Zbl's XML	Provides checked and rich article metadata given ZblID
PdfJbIm	PDF as file or input stream	Re-compressed PDF as file or in output stream	Reduces size of PDF files containing images by about 30%
Pdfsizeopt	PDF as file (content/ <file name>)	Optimized PDF as file	Reduction of PDF size by further thirty percent if used after running **PdfJbIm**
math_metadata_lookup	Search options (title/author/year plain text strings)	Metadata found in mathematical reviews databases in desired format (plain text/ HTML/XML/Ruby/YAML)	Stand-alone tool able to search through mathematical reviews databases and fetch metadata according to given search options
External/standalone tools			
ME	Unorganized collection of scanned pages of documents	PDFs of digital publications organized to collections of journals/volumes/articles etc. with metadata description	Stand-alone tool for organization and management of digitized publications able to prepare EuDML-ready full texts and metadata description that enables data providers without their own solution to participate in the EuDML project

4.1 Roadmap to Working Prototype

Analyzing subsystem components in Tables 2 and 3, it seems inevitable, because of reasons of efficiency, overlapping functionality, etc., that current eutools will have to be integrated into bigger programs — one per subsystem.

As a next step, eutools will be merged into bigger components (one enhancer per subsystem) and tested on the central EuDML site on real data from providers. The schedule to realize this will depend on the data collected and integration strategies developed by ICM. As a dozen programming languages, third-party tools and libraries must be collected and integrated, the scenario for integration is rather complicated — a set of rules for it has been written at `https://wiki.eudml.eu/eudml/EuDML_System_Documentation` for internal communication of developers.

4.2 Future Work

Before, and after, enhancement duplicate document items should be detected and/or merged. A strategy for metadata conflict resolution remains to be developed, e.g. when a paper is available from different sources and the same item has different/conflicting metadata, usually compared to those from Zentralblatt.

A policy might be as follows: when a metadata field is absent or empty in the provider's supplied metadata, and exists in the corresponding harvested Zbl record, then we should use the Zbl values. The paper might have keywords in various languages and lack English keywords: this is a typical added value/use case of enhancement expected from an enhancer, so the rule must take this kind of thing into consideration.

We could refine this policy on the basis of particular elements:

- title, source, author, main language, English title: use original if present
- MSC, English keywords: it could be added value to add those provided by Zbl if they do not match the original ones, even if the original ones exist! (that would make more links/associations between our articles).

These and other issues have to be decided before fully-fledged enhancing eutools reach their place in the EuDML system.

References

[1] ParaCite. http://paracite.eprints.org/.

[2] Digitization Metadata Editor, version 2.0 (Feb 11, 2011), February 2011. `http://sourceforge.net/projects/dme/`.

[3] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[4] American Mathematical Society. Mathematical Reviews, 2010. `http://www.ams.org/mr-database/`.

[5] Josef B. Baker, Alan P. Sexton, and Volker Sorge. Extracting Precise Data from PDF Documents for Mathematical Formula Recognition. In *DAS 2008: Proceedings of The Eighth IAPR Workshop on Document Analysis Systems*, 2008.

[6] Josef B. Baker, Alan P. Sexton, and Volker Sorge. Extracting Precise Data on the Mathematical Content of PDF Documents. In Petr Sojka, editor, *DML 2008: Proceedings of Towards Digital Mathematics Library*, pages 75–79. Masaryk University, 2008.

[7] Josef B. Baker, Alan P. Sexton, and Volker Sorge. A Linear Grammar Approach to Mathematical Formula Recognition from PDF. In *Proceedings of the 8th International Conference on Mathematical Knowledge Management in the Proceedings of the Conference in Intelligent Computer Mathematics*, volume 5625 of *LNAI*, pages 201–216. Springer, 2009.

[8] Miroslav Bartošek, Petr Kovář, Martin Šárfy, and Michal Růžička. Metadata Editor, 2010. http://is.muni.cz/publication/927548?lang=en.

[9] Thierry Bouche. CEDRICS: When CEDRAM Meets Tralics. In Petr Sojka, editor, *Proceedings of DML 2008*, pages 153–165, Birmingham, UK, July 2008. Masaryk University. http://dml.cz/dmlcz/702544.

[10] Chien-Chih Chen, Kai-Hsiang Yang, Hung-Yu Kao, and Jan-Ming Ho. BibPro: A citation parser based on sequence alignment techniques. *Proceedings of International Conference on Advanced Information Networking and Applications, AINA*, pages 1175–1180, 2008.

[11] JBIG Committee. *14492 FCD*. ISO/IEC JTC 1/SC 29/WG 1, 1999. http://www.jpeg.org/public/fcd14492.pdf.

[12] FIZ Karlsruhe. Zentralblatt MATH – ZBMATH Online Database, 2010. http://www.zentralblatt-math.org/zmath/.

[13] Free Software Foundation. GNU General Public License, 2010. http://www.gnu.org/licenses/gpl.html.

[14] Free Software Foundation. GNU Lesser General Public License, 2010. http://www.gnu.org/licenses/lgpl.html.

[15] José Grimm. Tralics, a LaTeX to XML Translator. *TUGboat*, 24(3), 2003. http://www-sop.inria.fr/apics/tralics/.

[16] Deepank Gupta, Bob Morris, Terry Catapano, and Guido Sautter. A New Approach towards Bibliographic Reference Identification, Parsing and Inline Citation Matching. In Sanjay Ranka, Srinivas Aluru, Rajkumar Buyya, Yeh-Ching Chung, Sumeet Dua, Ananth Grama, Sandeep K. S. Gupta, Rajeev Kumar, and Vir V. Phoha, editors, *Contemporary Computing*. Springer Berlin Heidelberg, 2009. http://dx.doi.org/10.1007/978-3-642-03547-0_10.

[17] Radim Hatlapatka and Petr Sojka. PDF Enhancements Tools for a Digital Library: pdfJbIm and pdfsign. In Petr Sojka, editor, *Proceedings of DML 2010*, pages 45–55, Paris, France, July 2010. Masaryk University. http://is.muni.cz/publication/891674/.

[18] Radim Hatlapatka and Petr Sojka. Recompression of Bitmaps in PDF using JBIG2 format, December 2010. http://is.muni.cz/publication/927601?lang=en.

[19] Erik Hetzner. 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*, pages 280–284, New York, NY, USA, 2008. ACM. http://doi.acm.org/10.1145/1378889.1378937.

[20] Google HP. Tesseract. http://code.google.com/p/tesseract-ocr/.

[21] Michael Jost, Thierry Bouche, Claude Goutorbe, and Jean-Paul Jorda. The EuDML metadata schema, November 2010. Deliverable D3.2 of EU CIP-ICT-PSP project 250503 EuDML: The European Digital Mathematics Library, http://eudml.eu/.

[22] Petr Kovář. math_metadata_lookup, 2011. https://github.com/pejuko/math_metadata_lookup.

[23] Adam Langley. Homepage of jbig2enc encoder. [online]. http://github.com/agl/jbig2enc.

[24] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710, 1966.

[25] PDFBox.org. PDFBox. `http://www.pdfbox.org/`.

[26] Peter Szabó. Pdfsizeopt. `http://code.google.com/p/pdfsizeopt/`.

[27] Michal Růžička and Petr Sojka. Scientific Journal Processing System with the Capability of Exporting to a Digital Library using MathML, December 2010. `http://is.muni.cz/publication/927596?lang=en`.

[28] Petr Sojka, Josef Baker, Alan Sexton, and Volker Sorge. A State of the Art Report on Augmenting Metadata Techniques and Technology, December 2010. Deliverable D7.1 of EU CIP-ICT-PSP project 250503 EuDML: The European Digital Mathematics Library, `http://eudml.eu/`.

[29] Petr Sojka and Radim Hatlapatka. Document Engineering for a Digital Library: PDF recompression using JBIG2 and other optimization of PDF documents. In *Proceedings of the ACM Conference on Document Engineering, DocEng 2010*, pages 3–12, Manchester, September 2010. Association of Computing Machinery. `http://portal.acm.org/citation.cfm?id=1860563`.

[30] Charles Sutton and Andrew McCallum. *Introduction to Statistical Relational Learning*. MIT Press, 2006.

Index