D-EE4.2d FUNCTIONAL EXTENSION PARSER | EE4
Final Prototype

Due: (extra)  Actual: Month 53

Internal / external: External
Activity type: RTD
Participant number: 4
Participant short name: UIBK
Dissemination level: PU Public

Revisions

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Author</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Draft</td>
<td>Lukas Gander, Günter Mühlberger</td>
<td>9 December 2011</td>
<td>Created</td>
</tr>
<tr>
<td>1.0</td>
<td>Final</td>
<td>*</td>
<td>20 December 2011</td>
<td>Comments of internal review processed</td>
</tr>
<tr>
<td>1.1</td>
<td>Final</td>
<td>*</td>
<td>25 May 2012</td>
<td>New version</td>
</tr>
</tbody>
</table>

Approvals

<table>
<thead>
<tr>
<th>Version</th>
<th>Date of approval</th>
<th>Name</th>
<th>Role in project</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>14 December 2011</td>
<td>Ines Jerele</td>
<td>Internal reviewer</td>
<td>OK</td>
</tr>
<tr>
<td>0.1</td>
<td>20 December 2011</td>
<td>Clemens Neudecker</td>
<td>Technical Project Manager</td>
<td>OK</td>
</tr>
<tr>
<td>1.0</td>
<td>22 December 2011</td>
<td>Hildelies Balk</td>
<td>Project Director</td>
<td>OK</td>
</tr>
<tr>
<td>1.1</td>
<td>31 May 2012</td>
<td>Clemens Neudecker</td>
<td>Technical Project Manager</td>
<td>OK</td>
</tr>
<tr>
<td>1.1</td>
<td>31 May 2012</td>
<td>Hildelies Balk</td>
<td>Project Director</td>
<td>OK</td>
</tr>
</tbody>
</table>

Distribution

<table>
<thead>
<tr>
<th>Version</th>
<th>Date of sending</th>
<th>Name</th>
<th>Role in project</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>13 December 2011</td>
<td>Sven Schlarb</td>
<td>Internal reviewer</td>
</tr>
<tr>
<td>0.1</td>
<td>13 December 2011</td>
<td>Clemens Neudecker</td>
<td>Technical Project Manager</td>
</tr>
<tr>
<td>1.0</td>
<td>20 December 2011</td>
<td>Max Kaiser</td>
<td>SP EE leader</td>
</tr>
<tr>
<td>1.0</td>
<td>20 December 2011</td>
<td>Hildelies Balk</td>
<td>Project Director</td>
</tr>
<tr>
<td>1.0</td>
<td>23 December 2011</td>
<td>Liina Munari</td>
<td>EC Project officer</td>
</tr>
<tr>
<td>1.1</td>
<td>31 May 2012</td>
<td>Clemens Neudecker</td>
<td>Technical Project Manager</td>
</tr>
<tr>
<td>1.1</td>
<td>31 May 2012</td>
<td>Hildelies Balk</td>
<td>Project Director</td>
</tr>
<tr>
<td>1.1</td>
<td>1 June 2012</td>
<td>Liina Munari</td>
<td>EC Project officer</td>
</tr>
</tbody>
</table>
Table of Contents

1. EXECUTIVE SUMMARY 4

2. STRUCTURAL ANALYSIS OF DIGITISED DOCUMENTS 4
   2.1 Why we need structural information 4
   2.2 Information retrieval 5
   2.3 eBooks and text correction 6
   2.4 Navigation 7
   2.5 Display of text and reading order 8
   2.6 Export and download 9
   2.7 Print on Demand 9
   2.8 XML Export 12

3. SCIENTIFIC APPROACH: RULE BASED DOCUMENT IMAGE UNDERSTANDING USING A HYBRID FUZZY CLASSIFICATION SYSTEM 12
   3.1 State of the art 12

4. RESULTS 14
   4.1 Evaluation approach 14
   4.2 Results for general features 15
   4.3 Results for table of contents 17

5. IMPLEMENTATION 18
   5.1 General considerations 18
   5.2 FEP Architecture 19

6. FEP CORE MODULE 20
   6.1 Why a rule engine? 20
   6.2 Module description 21
   6.3 Module functionality 22
   6.4 Interface to other modules 22
   6.5 Input / Output 22
   6.6 Restrictions 22
   6.7 FEPcontroller 22
6.8 Predefined rule set

7. VISUALISATION AND CORRECTION MODULE: FEP EDITOR

7.1 Module description
7.2 Views and tabs
7.3 Screenshots
7.4 Interface to other modules
7.5 Input / Output
7.6 Restrictions

8.1 Updated work plan
8.2 Main objectives for 2010 and 2011
8.3 Report on work carried out in 2010

9. REFERENCES
1. Executive summary

In work package EE4 we deliver the prototype of the Functional Extension Parser (FEP) which is a document understanding software that is capable of recognising basic structural features of digitised documents, namely historical books. Such features are e.g. page numbers, running titles, headings, footnotes, and table of contents entries. The FEP builds upon the results of OCR engines and provides METS/ALTO files as major output format. The results of the structural analysis and recognition are satisfying and in general beyond state-of-the-art methods. The FEP consists mainly of two parts: Firstly, a system that can read OCR output files, process them according to a given rule set, and transform them into enhanced formats. Secondly, a graphical user interface (GUI) that allows displaying, editing and correcting the results of the automated analysis.

In 2010, the FEP has been applied to new datasets, improved in terms of scalability and performance, integrated with the Impact Interoperability Framework and complemented with a correction interface. The current release provides a first version of a production tool ready to be used for the structural annotation of digitised documents. The actual implementation into a “real world” scenario has been done in cooperation with the EOD Network service within the extension of the IMPACT project (s. report on the extension).

2. Structural analysis of digitised documents

2.1 Why we need structural information

Optical Character Recognition is nowadays carried out for two main reasons: Firstly, the text can be used for information retrieval, namely full-text keyword search. Even if some words may contain spelling errors and will therefore be missed in a full-text search, this feature is highly attractive to readers and a “must-have” for digital libraries. Secondly, the raw text might serve as an intermediate result which can be the starting point for some manual spelling correction. Especially researchers who work with a book intensively are appreciating OCR for supporting their editorial work, e.g. by doing text correction and mark-up in TEI (Text Encoding Initiative). But also service companies which are converting scanned images into highly structured eBooks are in some cases using the raw text for their work.
In the same way as structural information already fulfilled a number of purposes in the analogue world it will also be a key feature for user-friendly applications in the digital world. Actually there are many ways to use structural information. The following sections provide some cursory considerations on this issue.

2.2 Information retrieval

As already indicated, one of the main applications in digital libraries is full-text search. Whereas today only the text is typically fed directly into a database, with structural information available it is possible to set up more specific services. Structural information can be used for improving the ranking of documents, e.g. headings could be ranked higher, footnotes could be ranked lower. A specific issue are running texts which repeat the title of a book, or the title of a chapter at every page. Terms being part of a running title distort the actual word count of simple full-text retrieval. Moreover a reader will be misled and will need to examine a number of search results that do not match his actual interest. With the information that a word appears in a running title (header), information retrieval is able to avoid this kind of noise and to deliver more meaningful search results.

Fig. 1 shows how simple full-text search could be improved by structural tagging: A search for “musketeers” in Hathi Trust retrieves a number of hits within a book, but if
we analyse them we can see that only one (green arrow) comes from the running text, whereas one (red arrow at first place) comes from the table of contents and three (red arrows below) come from the running title of the chapter.

Similar to this example is the use of structured full-text for facetted search. Facetted search offers the possibility to restrict search in a meaningful way by using metadata to narrow down an initial search result. Current digital library applications typically offer facets derived from the descriptive metadata of a record, e.g. facets are creator, year of publication, language, keyword, etc. With structural information available, additional facets can be offered, e.g. running text, headings, table of content pages, captions or footnotes. This would mean that a user could narrow down the results to hits found only in the running text, or only in footnotes.

Apart from ranking and faceting, also sorting has to be mentioned. Again a reader might be interested to sort a long list of hits according to structural features in order to get a better overview on the distribution of a search term within the document collection.

2.3 eBooks and text correction

One of the main research fields in IMPACT is the post-correction of raw OCR text. Currently, solutions from IBM as well as LMU are processing the complete text of a book without taking into account structural information. But obviously some users might have specific interests: They might be only interested in correcting footnotes, or correcting headings and caption lines of graphical elements, or just focussing on the body text. On the basis of structural information, the user might be able to set priorities as to which sections or features he might correct and therefore carry out a “selective text correction”.

This might also be the starting point for generating “real” eBooks, e.g. in EPUB or Mobipocket format. The transformation of raw OCR text into e.g. the EPUB format is rather simple from a technical point of view, but the main challenge is again correctly representing the structural information of documents using the eBook format. Though a number of service providers are offering this conversion process, there is no product on the market which offers a standard workflow for OCR, document understanding and correction of the text as well as the structural values in order to create accurate eBooks in a variety of formats.
2.4 Navigation

The automated detection of page numbers is one of the simpler tasks in document understanding and can be done with high accuracy. The page number is a sine-qua-non condition in digital libraries where users want to navigate directly to a given page.

One of the most important tools for navigation within books is the (original) table of contents (ToC). Also this structural element as well as single entries within a ToC can be detected with satisfying accuracy rates. Together with page numbers and headings within a book, a hyperlinked ToC can be offered to the reader where he clicks on the original image and is taken directly to the referenced heading or page. Also the text can be used to build a “new” ToC on a textual basis. The following screenshot shows the two features in Google Book Search (GBS).
This screenshot also makes clear that linking the original ToC with page numbers is a much “easier” task than building a correct textual ToC. As one can see there are several headings missing in the text list or wrongly interpreted. Nevertheless, though there are recognition errors it is still a highly valuable feature that supports readers in navigating through the document in a natural way. For future digital libraries this feature can be regarded as a “must-have”.

2.5 Display of text and reading order

In the case of 20th century books where OCR produces excellent results, also the text itself and not only the image of a page might be displayed to the user. When formatting this text, structural information can be used in many ways. Headings can be displayed with large fonts, footnotes with smaller fonts, column titles, page numbers etc. can be suppressed so that users are not disturbed in reading. This will be of high interest for mobile devices, where the screen is small and users want to scroll within a text without getting disturbed by running titles or page numbers. The following screenshot shows that without structural analysis the OCR engine is not able to correctly provide the reading order.
The yellow marked text are marginalia notes and footnotes which destroy the reading order of the text and are especially annoying for reading on mobile devices. With successful document understanding these elements would get the correct structural value and could then be displayed in a more adequate way to the user than it is currently the case. As indicated above, the text itself contains only very few spelling errors (marked in red) so that the reader will have a convenient reading experience.

### 2.6 Export and download

Users do not only want to use documents online, but do also want to work offline with them. Therefore the PDF format is one of the very popular features of digital libraries. In many cases the PDF assembles just the images of a book, but if the structural values are known, an enriched document can be provided to the user. Similar to features discussed above the ToC might be clickable, or all pictures and photos might be accessible via bookmarks. Structural metadata will also allow building new services, such as downloading just the ToC pages of a collection of documents, or just the pictures of a book with the captions. In another example, a user might carry out research on prefaces of documents (which are often a meta-statement of the author and therefore have a rather different structural value than the body text) and wants to download all prefaces of a given collection.

Another use case might be a user who wants to work with the text of a book and edit the text in text processing software. In this case the user is neither interested in the title page, nor in the table of contents, nor in column titles, nor in page numbers or signature marks, but only in the running text, the headings and footnotes. Structural tagging will allow him to receive a “clean” text containing the “intellectual property” of a book without contaminations provided by structural elements which were introduced just because the book was distributed in an analogue world.

### 2.7 Print on Demand

Very similar to future export and download functionalities of digital libraries are delivery formats which are already on the market. Mainly PoD (Print-on-Demand) and eBooks need to be mentioned.

In the case of PoD one would not expect that structural features play an important role since actually the scanned image is the basis for the printing process. But also in this case the decoding of the layout has some benefit over common methods: After scanning a book, a characteristic flipping of the text space can be observed. This is the case since the mechanics of books are everything but simple and therefore by turning
the pages, the position between a page and the scanning device significantly changes. When a page is cropped some margins are left but these margins are arbitrary. With usual image processing software the centring of the text space – which is the prerequisite to be able to print a nice looking book – will discover a number of challenges which cannot be overcome in a simple way. E.g. pages with only very little text on it, or pages with marginalia destroy the actual centring and need to be corrected by hand.

But as a matter of fact there is a structural element which allows to handle this problem, and which played an important role in book printing: the print space. It defines the position of all elements on a page and is the default value for the size of the margins. Therefore the recognition and reconstruction of the print space helps to easier produce PoD books. The following figure 4 illustrates the appearance and reconstruction of the print space.

In contrast to all other structural elements the print space cannot be directly “seen” on the page, it does not actually appear visible on the page. In contrast it is a grid which was used by the printer. Therefore it needs to be reconstructed on the basis of the printed elements of a page. The illustration below shows the assumptive print space in red. Especially for pages which contain only a few textual elements the reconstruction is not trivial:

One has to understand that the size of the page was always calculated on the basis of the print space. The print space contains the text of a book apart from margins,
signature marks, catch words and page numbers. The idea behind the print space was firstly to have a fixed frame to place the textual elements and secondly to have a point of origin from where the margins were calculated and thirdly it also was the measure for examining the “elegance” of the page.

Moreover a page never was seen on its own but always considered being one part of a pair. The calculation of a frame was therefore based on a pair of pages as it is shown in the figure below.

![Fig. 5 Reconstructed margins on the basis of the golden section](image)

The size of the margins was variable and could follow the golden section as in the example above but could follow other schemas as well. Actually the golden section was used rarely since the margins became so large that it was understood as a waste of paper. Derived from these considerations it is important to detect and reconstruct the print space for every single page in the book in order to allow the generation of nicely looking PoD copies. Document understanding would in this case provide the basic data for further generating a preprint file that can be sent to a Print on Demand service provider such as Amazon.
2.8 XML Export

In order to be able to use the structural information in other applications it is necessary to provide rich XML files. In order to support library standards, METS and ALTO where chosen as output format. Moreover, enhanced PDF files can be delivered as well.

3. Scientific approach: rule based document image understanding using a hybrid fuzzy classification system

3.1 State of the art

Our assumption is that similar to alphabets, which changed in time, but still can be understood by most people, also the layout of books will contain some general encoding that can be understood universally. Even if we do not speak the language of a book, we do in many cases understand, what the title page is, the table of contents, where there is a chapter heading, a running title, a page number or a footnote. Our work therefore concentrated on these universal layout features.

Even though the field of document understanding has not received as much attention as the research field of document analysis (layout segmentation and OCR), several methods of logical structure analysis have been proposed. Apart from the fact that most of them vary in the choice of the analysed document type (newspapers, scientific papers, books etc.), they also differ in the application of the theoretical model, in the choice of the algorithms used and in the selection of the used features of documents.

Regarding the application of the theoretical model, the research summaries of Chen (2007) and Mao (2003) make the distinction primarily between rule-based and grammar-based approaches. Rule-based representations are powerful and are therefore the model applied most frequently in relation to logical structure recognition. The disadvantages of rule-based approaches lie in their complexity with regard to the generation of rules as well as computing time required. These systems frequently develop a life of their own, with the result that forecasting expected results proves to be rather difficult.

A rule-based system generally consists of a fact base, a rule base and an inference machine, which applies the rules defined in the rule base to the facts of the fact base in order to obtain new facts. Most of the rule-based approaches are characterised by the knowledge specific to the application (domain knowledge) where experts are needed to code the rule base manually. Rule-based approaches applying expert knowledge were
successfully used by Niyogi et al. (1995), Kim et al. (2001), Klink et al. (2000), Lin et al. (1997) and by Ishitani (2005). The approaches of Niyogi, Kim and Lin are based on the classical Boolean logic whereas Klink et al propose a fuzzy logic approach.

Manually generated systems of rules are characterised by a static behaviour. This means that their intended purpose is limited to the designated area of application. They are not capable of reacting appropriately to new challenges and, by implication, cannot be adapted for a new area of application. For this reason there have been numerous efforts to develop systems capable of deriving rules automatically from manually annotated training data. In order to use statistical learning methods it is necessary to generate a large and representative quantity of annotated training data that can be analysed by self-learning methods. The problem here is that this data cannot be generated easily and is therefore associated with high costs. Systems built on a self-learning system are the methods of Laven et al (2005) and Esposito (2000).

Chen (2007) characterises document features using the following two categories:

- Image features (density of black pixels, number of horizontal lines in a segmented block)
- Structural features (relationship between objects on a page, e.g. distance between lines, height and width of lines, indent of lines etc.)

In our approach we are using a hybrid fuzzy classification system to map the physical structure of documents (blocks, paragraphs, lines), which will be taken from OCR, with specific logical elements. The classification system should be able to detect the following key features from documents such as:

- Page numbers
- Table of Contents
- Page headers
- Headings
- Footnotes
- Signature marks

We combined several approaches which were published within the research area of document understanding and are therefore following a hybrid approach. The core of our approach is a rule based system using fuzzy logic. Due to the nature of the digitisation process we believe that a fuzzy logic approach also proposed by Klink et al [9] is more appropriate than classic Boolean logic. During the scanning process several flaws may occur, which make it difficult to decide, for example, if an indent of a line is “normal” or “big”. Scanned pages are often skewed or the scanning panel does not
always have the same distance to the digitised document page, which results in the consequence that the definition of “default” cannot be formulated precisely. Fuzzy logic can handle such uncertainty.

Regarding the usage of a theoretical model we propose to use hand coded domain knowledge rules as well as learned rules using a learning algorithm. This hybrid approach is new since all published approaches are either using one of the models but not both. The reason for this decision is that we believe that some of the listed logical elements can be found easily by defining a set of hand coded knowledge based rules. The layout of page numbers or captions is well known and does not have to be derived exhaustingly by a learning algorithm. Page numbers, for example, are always placed either on top or at the bottom of a page. Regarding the horizontal alignment page numbers have the characteristic that they are either centred or alternating on the left and the right side of the page. For other logical structure types such as headings, footnotes and normal text, the domain knowledge is not that obvious. The rules for such structure types will be generated from a set of labelled data. Another advantage of the decision to limit the number of structure types for which the rules are generated automatically is that the selection of the features, which serve as input for the learning algorithm, is eased. We use a variation of the Wang&Mendel algorithm (1992) to generate the fuzzy rule base for some of the logical elements.

Our complete approach is described in a paper provided by the project team [Gander 2011] and has led to a patent application at the European Patent Office.

4. Results

4.1 Evaluation approach

For evaluation purposes, 200 books from different institutions were selected and annotated manually by a service provider. The dataset consists of documents from the Bibliotheque nationale de France, several members of the eBook on Demand Network and some samples from our own repository. The dataset was divided into two distinct subsets. The first subset consists of 160 books and was used as a training set. All rules are learned by using exclusively the annotated data samples from the training set. The second subset consists of 40 books and is used for evaluation purposes only, which means that the learning algorithm does not use the annotated ground truth data from this subset. The division into the two distinct subsets was done randomly in order to guarantee a representative selection of the evaluation set.
The fragmentation of the dataset has mainly two reasons:

1. The learned rules from the training set should be applied to a dataset which did not influence the rule learning process, in order to show how well the learned rules perform on previously unknown data samples.
2. The comparison between the results of the training- and the evaluation set gives a good indication about the quality of the learned rules. If the quality measurement of the evaluation set is close to that of the training set then it can be stated that the results are representative. On the contrary, if the quality of the evaluation is much worse than the quality of the training set, then it may be the case that we can observe the over-fitting effect and that therefore the learned rules are not appropriate for new and unknown data samples.

### 4.2 Results for general features

The results of the training and evaluation sets are outlined in fig. 6 and fig 7.

<table>
<thead>
<tr>
<th>Structure Type</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page number</td>
<td>0.97</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>Page header</td>
<td>0.97</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>Signature mark</td>
<td>0.67</td>
<td>0.91</td>
<td>0.77</td>
</tr>
<tr>
<td>Text</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>Footnote</td>
<td>0.93</td>
<td>0.91</td>
<td>0.92</td>
</tr>
<tr>
<td>Heading</td>
<td>0.87</td>
<td>0.81</td>
<td>0.84</td>
</tr>
</tbody>
</table>

**Fig. 6 Results training set**

<table>
<thead>
<tr>
<th>Structure type</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page number</td>
<td>0.97</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>Page header</td>
<td>0.88*</td>
<td>1</td>
<td>0.94</td>
</tr>
<tr>
<td>Signature mark</td>
<td>0.68</td>
<td>0.89</td>
<td>0.77</td>
</tr>
<tr>
<td>Text</td>
<td>0.99</td>
<td>0.98</td>
<td>0.98</td>
</tr>
</tbody>
</table>
Comparing the results from the evaluation and the training sets several observations can be made. Structural elements which are detected by encoding domain knowledge into rules (page numbers, page header and signature mark) do have nearly the same quality within the training and the evaluation set. The only remarkable difference occurs in the recall regarding page headers (marked with "**" in the table). The reason for this can be explained easily: Within the evaluation set there are two books where the order of the images is corrupt. There are therefore parts where the page numbers do not occur alternating at the left, respectively right side. The reconstructed pagination contains errors which influence the detection of the page headers. The domain knowledge that the page header is on the right hand side of the page number for even numbers and on the left hand side for odd numbers does not apply in this case. Due to the much smaller size of the evaluation set such statistical runaway values have a much higher impact on the quality of the results than in the training set. Beside the recall of the page header there is nearly no difference between the quality measurements of the structural elements detected by domain knowledge based rules. This fact indicates that the rules are representative and the quality measurement is quite reliable.

The comparison of the quality measurements regarding the structure types, which are detected by using a learned set of rules (text, footnotes and headings), does also not differ very much between the two sets. Also in this case we have one noticeable difference between the training and the evaluation set. The recall of footnotes (marked with "***" within the table) differs very much from the recall within the training set. The reason for this difference can also be explained by two statistical runaway values within the much smaller evaluation set. If the statistical runaway values are removed from the evaluation set, the recall for the footnotes within the evaluation set would be very close to the corresponding number within the training set.

Summarizing the results it can be stated that the results are convincing, especially since there are still some optimisation strategies available which are currently not implemented. Especially concerning the quality measurements regarding the headlines we expect much better results by implementing a further refinement step which tries to detect the table of content and compares the TOC entries with the actual results.
Furthermore it has to be noticed that the goals defined within the document of work are already fulfilled.

4.3 Results for table of contents

In contrast to the features described above where comparable figures are missing the extraction of table of contents pages and table of contents entries is part of the INEX competition. Our original plans where to take part in the INEX 2011 competition, but the time constraints did not allow this. Nevertheless we were able to get the INEX dataset for training and evaluation purposes and to develop a rule set to detect the table of content pages within the books and a rule set to reconstruct the single table of content entries and link them within the pages in the books. The results, outlined in fig. 8 and fig. 9, are based on the evaluation dataset of the INEX competition in 2011 [Doucet 2011]. Concerning the evaluation of the dataset of 2011 we identified two important measurements.

Figure 8 outlines the results concerning the quality of the detection of the table of content pages within the dataset.

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>True positive</td>
<td>1138</td>
<td>91.4</td>
</tr>
<tr>
<td>False negative</td>
<td>107</td>
<td>8.6</td>
</tr>
<tr>
<td>False positive</td>
<td>124</td>
<td>9.8</td>
</tr>
<tr>
<td>Total</td>
<td>1245</td>
<td>100</td>
</tr>
</tbody>
</table>

**Fig. 8 Table of content detection**

The INEX 2011 dataset contained a total number of 1245 table of content pages. We identified 1262 pages within the dataset as ToC pages. Figure 8. shows that 1138 ToC pages were assigned correctly, 107 ToC pages were not detected by our system and 124 pages were identified misleadingly as ToC pages. These numbers lead to a precision of 0.901, a recall of 0.914 and an f-measure of 0.907. Results concerning the ToC page detection were not published and hence are not comparable to our results.

Figure 9. shows the results concerning the reconstruction of the table of content entries. The evaluation was done using the XRCE link-based measure. The XRCE link–based measure permits evaluating the performance of the systems by matching ToC entries primarily based on links rather than titles [Doucet 2011]. It can be seen that our results are slightly better to those of the winner of the 2011 competition.
### Institution Performance

<table>
<thead>
<tr>
<th>Institution</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPACT-FEP</td>
<td>0.682</td>
<td>0.727</td>
<td>0.662</td>
</tr>
<tr>
<td>Microsoft Serbia</td>
<td>0.702</td>
<td>0.645</td>
<td>0.651</td>
</tr>
<tr>
<td>Nankai</td>
<td>0.674</td>
<td>0.676</td>
<td>0.632</td>
</tr>
<tr>
<td>Xerox</td>
<td>0.551</td>
<td>0.759</td>
<td>0.581</td>
</tr>
<tr>
<td>GREYC</td>
<td>0.499</td>
<td>0.652</td>
<td>0.507</td>
</tr>
</tbody>
</table>

**Fig. 9 TOC- entry reconstruction**

### 5. Implementation

#### 5.1 General considerations

State-of-the-art OCR engines, such as the ABBYY FineReader actually "know" much more about the image and the text than one sees at first glance by receiving the simple text output. Within the recognition process an advanced analysis is carried out on the macro as well as the micro level of images and pages. As a result, OCR engines typically provide or keep data on the following document and text features:

1. Coordinates of regions and blocks
2. Types of blocks, e.g. text, picture and table
3. Coordinates of lines, their left and right indent, their baseline, their font size, etc.
4. Languages used in the document, words and if they were found in the internal dictionary of an OCR engine
5. Coordinates and confidence of each single character including formatting information, such as bold, italic or superscript

This wide array of information can be accessed either via a standard API during the recognition process itself, or via an XML output file which is provided for every single page together with the actual text file containing the full-text of the document.

The FEP team decided to build the module using the XML output file for several reasons:

1. FEP will be independent from a specific OCR engine.
2. FEP is also independent in the sense that the transformation is done during the recognition process itself, but the FEP enrichment can be done anytime after having finished the actual text recognition process.

3. Last but not least the enrichment process can be repeated several times on the same XML output file. E.g. documents which were too complex and difficult to process might be processed again on the basis of an improved transformation. Furthermore, other transformation processes might be carried out on the same XML output file which have not been taken into consideration at the beginning.

The main approach of FEP is therefore to set up a generic infrastructure where OCR XML output files can be transformed into enriched files serving the requirements as they are appearing for digital library applications in a wider sense.

5.2 FEP Architecture

The FEP consists of six main components:

1. Ingest Module: The ingest module receives images and OCR XML Files. The main functionality is to run an XSL transformation so that the original OCR XML files are transformed into the FEP XML instances. The FEP XML instances are than passed on to the database.

2. Server File System: This module manages the images and OCR XML files as well as configuration files for the rule engine.

3. XML DB: The XML DB serves as the “knowledge base” for FEP. It contains the FEP XML instances and receives the new facts from the rules engine. Moreover it provides these data to the visualization and correction as well as to the export module.

4. FEP Core: This module includes the rule database and a controller. Here, the predefined rules are applied and new facts (=results) are derived. JESS is used as the rule engine.¹

5. Visualization and correction module: The user will be able to visualise and correct results within the visualisation and correction module. It is accessed via a web-based GUI.

6. Export module: The export module offers the data in several ways: Results are either saved in an updated FEP XML instance, or they will be transformed into new XML instances, of another XMLSchema, like e.g. METS/ALTO for digital library applications, or eBook formats such as EPUB or results are applied to the images themselves in order to support print on demand requirements.

The modules are written in JAVA and are interacting with each other via web-services. The visualisation and correction tool is web-based and using the Google Web Tool Kit.

¹ http://www.jessrules.com/
The decision for GWT was taken since it is a rich and powerful web framework based on JavaScript and Servlets. Last but not least it is cross-browser compatible.

The following data flow model explains how the components work together.

In the following section the two main components are described in more detail: The FEP Core module and the FEP Visualisation module.

6. FEP Core module

6.1 Why a rule engine?

The second issue was to take a rule engine, in our case JESS. The reasoning in this case is based on implementation aspects. The pros and cons of a rule engine are listed very well at the supporting documents for JESS and have been taken seriously into
account by the FEP development team: ² Rule engines are recommended under the following conditions:

- **If there is a considerable amount of “decision making”, e.g. if there are more than three conditions in a rule or in other words if rules tend to become complex.**
  From previous projects we do know that large “rule based” knowledge bases are difficult to maintain because it might be difficult to keep track of the inter-dependencies between rules.

- **If the algorithm is dynamic, e.g. changes occur rather often.**
  Although within the IMPACT project we are focusing on monographs, it is obvious that – given that the final prototype shows the overall feasibility of our approach – other document types, such as journals, dictionaries, handbooks, brochures, etc. may also be processed with FEP. These documents will require adaptation of existing rules and introduction of new rules. This consideration comes along with another aspect: Maintenance of a set of rules for the long term will be eased by using a rule engine since they are stored separately in an XML file.

- **Last but not least, a rules engine can be used as a “prototyping” and “simulation” tool, separating “business logic” from the rest of the system, aiding in the effort to clearly “separate concerns”.**
  In other words: Since the final product of EE4 will be a research prototype it might be the more feasible way to use this prototype for developing the best rules for a given problem and then implement these (tested and evaluated) rules within another software platform.

Working with the rule engine during the “proof of concept” phase fulfilled our expectations. Especially the communication between technical developers and domain experts is eased on the basis of rules which can be easily explained and discussed within the team.

### 6.2 Module description

The module itself is implemented as a Java web service (using the Apache Axis2Java framework). It is called **FEP Core**. This web service will be installed on a web server at the University Library of Innsbruck. It is online accessible, but the address may change

---

during the course of the project. For reviewing purposes it will be certainly supported resp. updated.

6.3 Module functionality

The input consists of a database book identifier and a configuration XML file, in which parts of a set of FEP XML instances or an already created set of JESS facts gained from a previous enhancement cycle can be declared. The configuration XML also defines a set of rules that will be applied on the declared facts, and also an output selection section, where the final JESS facts are declared for preservation. The rules themselves are written as Constraint Logic Programming (CLP) files and will be adapted and improved during the whole project. Every enhancement consists of a minimum of two rule assignment steps, a detection step and a reconstruction step.

6.4 Interface to other modules

FEP Core may be called via the visualisation and correction module over WSDL/SOAP Web Services.

6.5 Input / Output

Input: Book Identifier, Configuration XML, FEP XML

Output: Enhanced FEP XML

6.6 Restrictions

The access to the web service is restricted to a specific user, identified by a username and a password. The username as well as the password are only available to partners of the project. Furthermore, there are no different roles and the valid user has full access to the functionality of the web service. The service follows established WS-Security principles.

6.7 FEPcontroller

The FEPcontroller collects all the data declared in the configuration XML and hands them over to the JESS Engine. Then it reads the declared final facts from the rule engine and stores them in the XML Database. But the controller does not only hand over data between the FEP Core parts, it can be told to hand over all the data fragmented to gain control over the performance.
6.8 Predefined rule set

One of the core parts is the rule set that allows the identification of specific entities within an FEP XML (like page numbers for example) and to reconstruct them. To make the system as flexible as possible, these rules will not be part of the source code. The JESS Rule Engine takes the base facts, which are extracted from the existing FEP XML, and applies the given rules to detect the desired information. The newly generated facts are the basis for the reconstruction step which is once more a JESS rule engine cycle with reconstruction rules. Based on the results of the engine, the enhanced FEP XML is created.

Additionally, there is the possibility to transform the given rules and facts to RuleML which is developed “as the canonical Web language for rules using XML markup, formal semantics, and efficient implementations.” Using a standardised format offers the possibility to use (and even to refine) the rules in other projects as well.

7. Visualisation and Correction module: FEP Editor

The visualisation module allows the user to have a detailed look at the text recognition results as well as at the enhanced results stemming from applying the rule engine. The correction module allows the user to correct the results.

7.1 Module description

The module itself is called FEP Editor and is implemented as a Java web application using Google Web Toolkit. The main purpose of this application is to have a tool that allows easy access to the FEP XML. The FEP Editor is also used as pre-stage and as a test utility for the correction part of the module. The main functionality of this module is in a very early development phase because the possibility to visualise the FEP XML facilitates the formulation of the rules for the detection of the pagination, print space and default body type.

7.2 Views and tabs

There are several views and tabs available. We can distinguish three main views:

a) Document Overview

3 Cf.: http://www.ruleml.org/
This tab provides statistical data about the document and allows users to orientate themselves.

b) Document Viewer

This tab provides access to all data of the document. Its purpose is again to inform the user. Though editing and correcting is possible this tab does not play a role in the editing and correction workflows.

c) Special Editors

If users want to edit and correct specific structural elements, such as page numbers, table of contents entries, and print space, dedicated workflows and editors are provided. The correction process is started from the Document Overview tab and leads the user through the correction workflow. The main idea is that all correction steps shall be self-explaining.

7.3 Screenshots

In the following we explain the main tabs via a series of screenshots.

Fig.11 FEP: Document Overview

In the Document Overview some basic data are provided to the user so that he can get a quick impression about the number of pages, the structural elements detected in the document, the status of the correction process, etc.
In the Document Viewer the user is able to display all data which are available for this document: OCR data (e.g. blocks, lines, ...), data from the layout recognition process (e.g. page numbers, headings, running titles,...) and – if available – data from the ground truth. This tab is not foreseen for actual correction, but as a comfortable way to access all the data connected with a book.
In the example above we see the (corrected) results for a table of contents.

As indicated in the introduction the editor pages can only be called by starting the correction workflow. Since it is expected that not all books and not all users will have the same requirements for correction, the user is able to select between several correction workflows.

One of the basic correction steps is to control the page numbers.

Green page numbers are those where the OCR has recognised some digits and which comply with the logical order derived from the numbers before and after. Numbers are shown in red if the OCR engine did not recognise a number of the page. Page numbers can be corrected in a quick mode.

In order to correct the print space several steps are necessary which will not be shown here in detail. The main steps are firstly to set the width and height of the print space and finally to apply it to all pages. The following screenshot shows the final step where the user has the possibility to adjust the position of the print space within the original image.
Fig. 15 Print space editor - final step

Another important step is to correct the table of contents. Here several modes are possible. One of the prerequisites for every analysis is to detect the table-of-contents-pages. Another step is to actually correct the table of contents entries. The following screenshot shows this step:

Fig. 16 Toc Entries Correction
On the left hand side we see the entries of a table of contents. According to a selected entry on the right hand side the linked page respective heading is displayed. In this way the user is directly able to visually control if the linking was correct.

Last but not least the hierarchy and the text of the table of contents entries can be edited and corrected as well. This is done in the following tab:

![Fig. 17ToC Hierarchy Editor](image)

### 7.4 Interface to other modules

The web service FEP Core can be called via the FEP Editor

### 7.5 Input / Output

Input: OCR XML, FEP XML

Output: Statistical information (XML format), enhanced FEP XML

### 7.6 Restrictions

The access to the web application is restricted to one specific user, identified by a username and a password. The username as well as the password are only available to partners of the project. Furthermore, there are no different roles and the valid user has full access to the functionality of the web application.
8. Mass digitisation

In order to test the performance of FEP within a mass-digitisation workflow we processed 2,000 books from the INEX 2009 and INEX 2011 dataset. The main focus of this stress test was to check if the system is able to run stable over a long period without serious break downs. The 2,000 books of the two INEX datasets were divided into four distinct parts (500 documents each) to simulate parallel processing. The table in Figure 18 outlines the most interesting facts about the applied stress test to proof Mass digitisation.

<table>
<thead>
<tr>
<th></th>
<th>No. docs</th>
<th>Processing Time (hours)</th>
<th>Docs aborted</th>
<th>Docs failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client 1</td>
<td>500</td>
<td>627</td>
<td>58</td>
<td>20</td>
</tr>
<tr>
<td>Client 2</td>
<td>500</td>
<td>599</td>
<td>24</td>
<td>9</td>
</tr>
<tr>
<td>Client 3</td>
<td>500</td>
<td>615</td>
<td>43</td>
<td>13</td>
</tr>
<tr>
<td>Client 4</td>
<td>500</td>
<td>603</td>
<td>29</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>2,000</td>
<td>2444</td>
<td>154</td>
<td>52</td>
</tr>
</tbody>
</table>

First of all it has to be mentioned that there didn’t occur any serious breakdowns of the system itself. The analysis of the 2,000 books was interrupted only once due to a power failure during a weekend. Other serious break downs don’t have to be reported.

The FEP was installed at four simple desktop PC's which served as processing stations. 500 documents from the datasets were assigned randomly to each processing station. Due to the fact that the behaviour of rule based system is very hard to predict because they tend to develop a life of their own we implemented a security mechanism which aborts the analysis with a timeout event if the analysis of a document exceeds 2 hours (listed in column Docs aborted). These documents were afterwards processed a
second time by increasing the timeout to 4 hours. Documents which also didn’t pass
the second time were then marked as failed and are listed in column Docs failed.

All in all it can be stated that the stress test proofed that the system would be capable
to handle thousands of books within a reasonable period of time. The stress test has
also shown that the system can be scaled linearly simply by increasing the number
of processing stations. The total time needed for the analysis of the 2,000 books was
2,444 hours which equates to an analysis time of a little bit less than 75 minutes per
document. The time needed for the FEP analysis is hence comparable to the time
needed for OCR processing. Another interesting fact to be mentioned is that the
system was able to do the analysis for 1,948 (97.5 %) documents of the test set. The
quality of the analysis results are outlined in the Figures 8 and 9.

9. References


2010.

Boghardt, Martin: Archäologie des gedruckten Buches. Hrsg. von Paul Needham in
Verbindung mit Julie Boghardt. Wiesbaden: Harrassowitz Verlag 2008

Bontcheva, K. et al: Using human language technology for automatic annotation and

architecture and performance evaluation, International Journal of Document analysis

Dowding, Geoffrey: An Introduction to the History of Printing Types. An illustrated
Summary of the Main Stages in the Development of Type Design from 1440 up to the
Present Day. An Aid to Type Face Identification. London: The British Library & Oak
Knoll Press 1961 (Reprint from 1998)

Esposito, Floriana; Malerba, Donato; Lisi, Francesca, Machine Learning for Intelligent
Processing of Printed Documents, Journal of Intelligent Information Systems, Volume


Gander, Lukas; Lezuo, Cornelia; Unterweger, Raphael: Rule based document understanding of historical books using a hybrid fuzzy classification system. HIP '11 Proceedings of the 2011 Workshop on Historical Document Imaging and Processing


http://www.dante.de/dante/DTK/dtk02_4/dtk02_4_kohm_satzspiegelkonstruktionen.html


