IMPACT is supported by the European Community under the FP7 ICT Work Programme. The project is coordinated by the National Library of the Netherlands.

Report on the comparison of Tesseract and ABBYY FineReader OCR engines

Marcin Heliński, Miłosz Kmieciak, Tomasz Parkoła

Poznań Supercomputing and Networking Center, Poland

Table of contents

1. Introduction

The aim of this report is to compare OCR accuracy of two well known OCR engines: Tesseract 3.0.1 and FineReader10 Corporate Edition. The comparison is based on Polish historical printed documents and ground-truth produced within the scope of the IMPACT project (available at http://dl.psnc.pl/activities/projekty/impact/results/). As a rule defined by the IMPACT project the document is assumed to be historical if it was printed before 1850. Tesseract and FineReader have been selected because both represent high-level solution coming from open-source and commercial community and both were the subject of research works in the scope of the IMPACT project. The idea of the works described in this report was to verify in practice how these two tools can be fine-tuned by the end user in order to improve recognition rate of the historical printed text as well as compare the results of OCR accuracy after fine-tuning of both tools.

In order to efficiently train Tesseract engine it was needed to enhance part of IMPACT ground-truth data with regions on a character level. Such a detailed ground-truth data could be then used to prepare appropriate training data for Tesseract engine. Training of FineReader was done using the built-in mechanism of this OCR engine and did require ground-truth data on a region level. For OCR accuracy rate calculation IMPACT ground-truth data on a region level was used. The tools used to calculate OCR accuracy were provided by the IMPACT project partner – the National Centre of Scientific Research "DEMOKRITOS".

There were several types of experiments performed in the context of OCR training, including document level training, whole dataset training and incremental OCR training. All of these experiments were done to provide an overall view of the OCR results that can be expected from particular OCR engines after particular training efforts. The conclusions include the observations related to the recognition accuracy on character and word level in the context of characteristics of the pages being OCRed.

The second chapter describes the training process applied to Tesseract and FineReader with details and rationale related to particular training approach. The third chapter includes main part of this report and is focused on the:

- Evaluation scenario and criteria that has been used in order to assess the recognition quality and further compare both OCR engines.
- Observations related to OCR recognition accuracy for particular documents processed with the use of certain OCR engine.
- Analysis of the recognition accuracy of the Tesseract and FineReader engines for the purpose of comparison.

The last chapter is a summary with conclusions related to the comparison of FineReader and Tesseract OCR engines, with the focus on the problems and challenges that certain OCR engine should face and improve.

2. Training process

In case of Tesseract automated approach to the training process has been selected. The training of the Tesseract covered all the necessary steps according to the guidelines of the

documentation. As a result a complete set of trained data for Tesseract was obtained and prepared for download for all interested parties. The trained data is available at http://dl.psnc.pl/activities/projekty/impact/results/. The trained data is available both for gothic and aniqua documents. It is important to note that these resources are constrained to the data selected during the IMPACT project, nevertheless they can be used for further analysis, research and improvements.

In case of FineReader the manual approach was selected. It means that the training process was performed using the built-in mechanism in the FineReader Corporate Edition. The training was prepared and performed according to the documentation and recommendations of the ABBYY company representatives (IMPACT project partner), who suggested to evaluate this feature of FineReader only on a document level, as this would guarantee good FineReader results. As also stated by the ABBYY, general training, e.g. for whole gothic or antiqua dataset, requires ABBYY paid involvement, which means internal training of the FineReader. As this paid service of training was not in the scope of the pilot work, this kind of results was not obtained for the FineReader. Nevertheless, in case of gothic text ABBYY provided estimated price for training FineReader on Polish gothic documents that are the subject of this report. The estimated rate could be then compared with efforts needed by PSNC staff to train Tesseract on the same set of gothic documents.

2.1. Tesseract training process

Tesseract OCR 3.0.1 can be fully trained in order to support non standard languages: character sets and glyphs. The training process is described in the training manual[1] and can be easily scripted to process training automatically (refer to train.sh script for detailed information). However, training data needs to be gathered and stored in a strictly specified form. In order to use the IMPACT dataset for Polish language containing set of historical images and ground-truthed corresponding full text content, it requires transformation. The transformation steps are described in the subsequent paragraphs.

Training data for Tesseract includes examples of scanned documents along with the full text content for these documents. As stated in the documentation, the training data should be as much natural as possible in the context of page layout, words spacing and characters combinations commonly used in the language. On the other hand, given images shouldn't contain too much noise (e.g. bad quality of scanned images) in order to provide valuable training data.

Full text content has to be stored in Tesseracts format called *box file*. It is made of character unicodes and boxes, that is coordinates of rectangles bounding all characters found in the scanned page. As a given page should base on a single font, it may be difficult to provide real historical documents as different font types are often used within the same page (e.g. italics). However, Tesseract allows characters to have a font type modifier attached, and therefore to distinguish glyphs of a different font type, but referring to the same character. This approach

[1] http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3

was used in the antiqua training process, where "$" character was added to indicate italic font type in the training data. In case of the training process for gothic texts, antiqua font was excluded from the training data because of two reasons. First of all the ratio between number of antiqua and gothic glyphs is low. Secondly, we wanted to test a vanila version of Tesseract with only one font trained.

In case of the IMPACT dataset, each page is represented by files that follow a particular naming convention. Files related to a single page are identified by a unique number, with a suffix depending on the file's type. File names suffixed by '.tif' contain images of scanned pages stored in a lossless TIF format. Full text content is stored in files complying with the PAGE XML format[2] (denoted by '.xml' suffix). Finally, documents from the dataset may contain both italic and non-italic or fraktur and non-fraktur font shapes (glyphs) in the same sentence, dataset is extended by further description. Each page may have a third corresponding file, a UTF-8 encoded list of identifiers of text regions that were selected as non-fraktur or italic words. These text files have '.txt' suffix.

In the first step glyph's repository needs to be created, which contains information on glyph shapes found in a particular page from the dataset. Information includes bounding coordinates of a glyph, unicode character mapping, binarisation threshold as well as font parameters. All this allows to generate a denoised form of the original page, e.g. with simplified page layout or excluded font types. To generate a glyph repository one needs to use a *Cutter* tool.

```
The following options are required:     --image     --xml
Usage: pl.psnc.synat.a12.aletheia.Cutter [options]
 Args:     directory name for output files
  Options:
        --font            mark generated glyphs as a {gothic, italic} font
        --help            print this help
  *     --image           file name of a image input
        --only-read-xml   reads only xml file, does not generate anything
        --tabu            file name of a text file containing list of noise
                          word's ids
        --tabu-types      list of TextRegion types to omit during processing
                          Default: []
  *     --xml             file name of a page xml input
```

This tool requires two files as an input: a page image and a ground truth XML file. In case of the IMPACT dataset, images should be transformed to PNG format prior to repository generation process. This can be done e.g. by using ImageMagick *convert*[3] tool:

```
convert 003287.tif 003287.png
```

Output repository should be stored in a separate directory, hence its name is a mandatory parameter of the application. The `--font` option allows to add font type to characters

[2] http://www.primaresearch.org/papers/ICPR2010_Pletschacher_PAGE.pdf

[3] ImageMagic identify tool described at http://www.imagemagick.org/script/convert.php

processed by the application, excluding characters from word regions listed in `--tabu` file. The latest are marked with 'noise' font type by default. In order to filter out text regions basing on their type, a list of tabu types can be passed by using the `--tabu-types` parameter. It is used for simplifying layout of pages generated in the second step. Available region types are:

```
paragraph, heading, caption, header, footer, page-number, drop-capital, credit,
floating, signature-mark, catch-word, marginalia, footnote, footnote-continued, TOC-
entry
```

For experiments described in this report only *paragraph* and *heading* region types were included as their font size is comparable and reading order is well defined. The latest parameter `--only-read-xml` enables dry run of the application, which allows to check if provided ground truth file contains any errors, e.g. referring to wrong characters encoding. Application can be run as in the following example:

```
java pl.psnc.synat.a12.aletheia.Cutter --image 003287.png --xml
003287.xml --tabu 003287.txt --tabu-types caption --tabu-types header
--font gothic 003287
```

which results in a glyphs repository in output directory named 003287/. The cutting process is as follows. Process text regions, unless a region's type was indicated as a tabu type. For each text region denoting words in a given ground truth XML file that does not appear on the tabu list, glyphs are processed. Each glyph is cropped from the original image basing on its bounding coordinates. For each cropped subimage a binarization threshold is sampled basing on the Min fuzziness approach implemented in the JAI[4] library. Eventually, binarised glyph is stored in a PNG file along with its unique identifier, coordinates within the original image, applied binarisation threshold, corresponding ground truthed code point and font attributes.

This tool was applied to pages from the IMPACT dataset ground truthed at the glyph level, that represent gothic and antiqua Polish historical texts. Processing summary is presented in the following table:

Table 1 Processing summary of the Polish historical texts

Dataset	Number of glyph repositories
Fraktur/Gothic	287
Antiqua	186

In the next step, glyphs repository can be used for generating denoised versions of the original images. As repository contains information on font types, one can filter out only required fonts, excluding e.g. words written as italic antiqua from the fraktur (gothic) texts. Moreover, as in the first step tabu regions could be excluded from the repository, a layout of the generated page can

[4]Java Adcanced Imaging API http://java.sun.com/javase/technologies/desktop/media/jai/

be simpler than in the original image. Page can be generated by *Generator* CLI (Command Line Interface) application:

```
The following options are required: -w, --width -h, --height
Usage: pl.psnc.synat.a12.generator.CLI [options]
Args: <box_dir_path> [ <image_filename> ]
  Options:
        --baselines      generate and output baselines offsets to given filename
    -b, --box            generate and output ".box" data to given filename
        --em-ratio       set space to em size ratio, used for spaces detection
        --font-type      output font type in output box file
  * -h, --height         set height of generated output image
        --no-image       do not generate output image
        --overlap        allows letter bounding boxes to overlap (default no
                         overlaping)
                         Default: false
        --skip-gothic    skip gothic letters in output
        --skip-italics   skip italic letters in output
        --skip-noise     skip noised letters in output
    -t, --text           generate and output text to stdio
    -v, --verbose        turn on verbose mode
  * -w, --width          set width of generated output image
```

Application requires four arguments: width and height of the generated image, path to glyphs repository and output image filename. In order to generate cleaned form of the original page, the first two arguments should match dimensions of the original image. These can be retrieved e.g. by a ImageMagick *identify* command[5]:

```
identify -format "-w %w -h %h" 003287.png
```

As font type filtering is required, one can use one of the `--skip-*` options. It is possible to generate *box* files related to cleaned images by passing the `--box` parameter along with requested file name, whereas the `--font-type` switch instructs application to attach encoded font type to characters stored in the box file. Eventually, exemplary application call:

```
java pl.psnc.synat.a12.generator.CLI -w 560 -h 1203 --box 003287-
clean.box --font-type 003287/ 003287-clean.png
```

reads glyphs repository from `003287` directory and generates both page image `003287.png` (with given dimension 560x1203) and corresponding box file `003287.box`. The latest contains font type marks e.g. for italic glyphs.

The generated image is made of binarised glyphs stored in a given glyphs repository. As coordinates within the original image are given for each glyph, the original glyph location can be easily restored, preserving letters, words and line spacing. Moreover, as glyph repository contains binarised images, generated images contain much less noise than the original ones.

[5]ImageMagic identify tool described at http://www.imagemagick.org/script/identify.php

Please note, that this approach represents local binarisation method and should be more effective than methods binarising whole image at once.

Finally, generated data can be provided for the Tesseract training process. This process can be automated with the following bash script[6] train.sh:

```
# Path to Tesseract instalation
TESS_HOME=tesseract-3.01
FONTNAME=antiqua

# paths with tesseract's binaries and shared data
TESS_BIN=$TESS_HOME/bin
TESS_SHARE=$TESS_HOME/share/tessdata

# document specific settings
LANG=pl

################################################

NAME=$LANG.$FONTNAME.exp
TIFNAME=.png

echo "combined 0 0 0 0 1" >font_properties &&
$TESS_BIN/unicharset_extractor *.box &&
for x in `cat files.txt`
do
       echo "tesseract training $x$TIFNAME"
        $TESS_BIN/tesseract $x$TIFNAME $NAME$x nobatch box.train.stderr
done
cat *.tr >combined.tr
$TESS_BIN/mftraining -F font_properties -U unicharset -O $FONTNAME.unicharset
combined.tr &&
$TESS_BIN/cntraining combined.tr || exit

mv pffmtable $FONTNAME.pffmtable
mv normproto $FONTNAME.normproto
mv inttemp $FONTNAME.inttemp

$TESS_BIN/combine_tessdata $FONTNAME.
# && cp -v $FONTNAME.traineddata $TESS_SHARE/$FONTNAME.traineddata
```

The transformation process for an exemplary page with id 003287 is summarised in the table below:

Table 2 Transformation process for an examplary page with id 003287

Transformation step	Input	Output
convert	003287.tif	003287.png
pl.psnc.synat.a12.aletheia.Cutter	003287.png 003287.xml 003287.txt	003287/
pl.psnc.synat.a12.generator.CLI	003287/	003287-clean.png 003287-clean.box

[6] This script was created for Tesseract 3.01 release.

train.sh	003287-clean.png 003287-clean.box	clean.traineddata

2.2 FineReader training process

FineReader 10 Corporate Edition allows training text recognition manually only on the document level. Only characters included in the alphabet of the recognition language can be used to train FineReader. Trained characters are stored in the user pattern which together with recognition language and other options can be saved into the file for later use. However, pattern can only be used for documents that have the same font, font size, and resolution as the document used to create the pattern.

Considering the above rules and limitations, the process of training FineReader began with creating OCR language. That language was based on the Polish language available in FineReader and was extended with the characters used in the training dataset. These contained mainly characters that cannot be entered from the keyboard, especially ligatures, old characters such as long s, and accented characters such as a with acute above. Two separate languages were created to support gothic and antiqua documents.

Once the recognition language was ready the pattern training could be started. The training process was performed on a cleaned black and white images. Training results were used for testing two data sets - cleaned black and white images and real scans without any changes.

For each training set the user pattern was created before training the first page had started. The user pattern name consisted of the associated training set name and the number of trained pages.

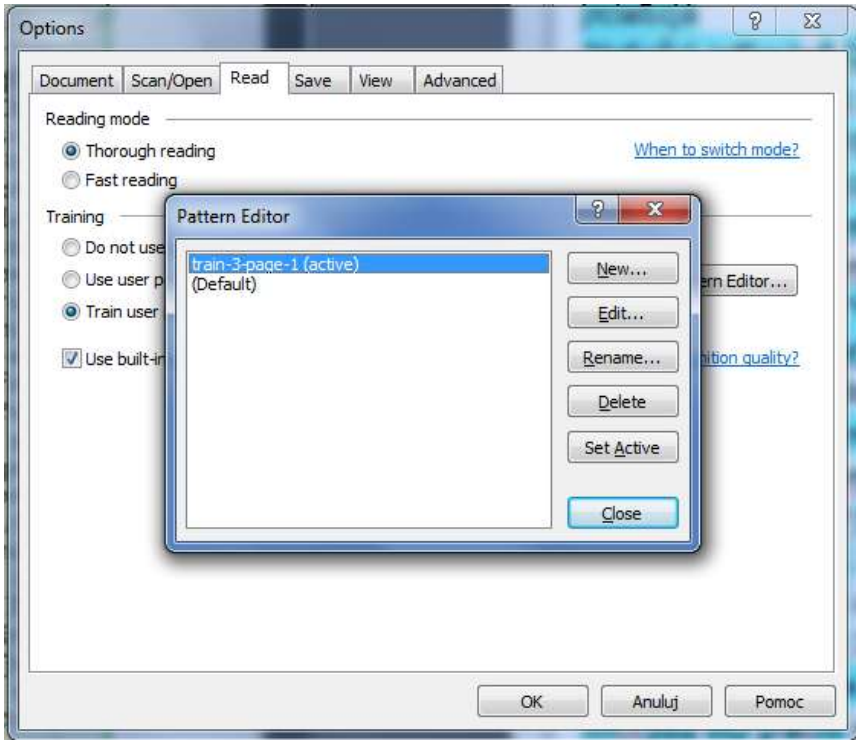


Figure 1 Pattern editor of the FineReader 10 CE OCR engine

While training, FineReader shows the unrecognized character bounded with the green frame. When the boundaries were detected incorrectly the frame size was manually adjusted to the character being recognized. If necessary, the character was entered from the keyboard or selected from the recognition language character set.

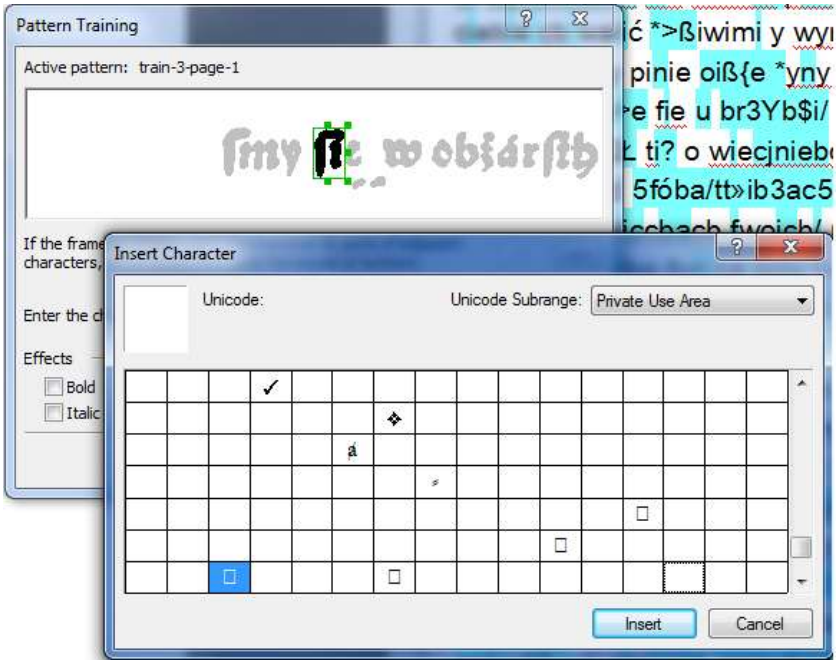


Figure 2 Training process of the FineReader 10 CE OCR engine

If the quality of the character being recognized was poor it was skipped on purpose. The reason for that was not to perturb the pattern with images showing part of a character similar to the other real character.

After training each page the pattern and the settings were saved to a file. Next the user pattern was renamed to include the following page. The above actions had been performed until all pages in the training set were trained.

The training approach used in the comparison is quite narrow and has several limitations described in the beginning of this section. Additionally it is very time-consuming. Unfortunately training FineReader on a broader scale requires purchasing an expensive service that exceeds the funds and scope of this pilot work.

During training several problems were encountered. The first problem was associated with the frame bounding the character being recognized. Sometimes it happened that the frame was not placed in the beginning of the character. The only way to adjust the frame size was to make it wider but the position of the left edge could not be changed. In that case the part of the character was skipped.

Another problem was with italic characters. Sometimes the character (especially "ſ" – long s) overlapped the character next to it. In that case the right edge of the bounding frame was placed in a way to avoid overlapping. This meant that only part of the character was within the frame but since all other cases were similar the recognized character could be stored in the pattern.

3. Comparison of Tesseract and FineReader

3.1 Evaluation scenario

In order to compare Tesseract and FineReader OCR engines, the evaluation scenario is based on the *hold-out* test and train approach, where dataset is randomly split into two disjoint subsets. First one is used only for the training purpose in order to create customised OCR engine and is referred to as training set. Later on, a customised OCR engine is applied to the second subset, a testing set. This results in full text content, of which quality is verified with the ground truth. The initial set of pages was divided into separate datasets, each referring to a single document, therefore enabling training process of FineReader (please refer to Section 2.2 for more details). Moreover, in case of Tesseract it was also possible to perform evaluation on a broader set of pages from several documents, hence additional 'all' dataset was created as well. The two following tables present number of pages used in datasets for training and testing for fraktur and antiqua fonts respectively.

Table 3 Number of pages used in datasets for training and testing for fraktur based documents.

Dataset name	Number of training pages	Number of test pages
test-14	4	14
test-16	5	17
test-19	4	14
test-21	4	15
test-22	9	41

test-3	6	22
test-all	59	228

Table 4 Number of pages used in datasets for training and testing for antiqua based documents.

Dataset name	Number of training pages	Number of test pages
test-na	28	108
test-oe	5	10
test-zr	7	28
test-all	38	148

The testing phase was conducted in two variants: cleaned test and real data test. In the first case test pages were transformed to so called cleaned versions. This was done in the same manner as for Tesseract training described in Section 2.1. The second case concerns original page scans containing much more noise. There are few reasons why such an approach was proposed. As this evaluation refers to characters and words recognition quality, it shouldn't be biased by additional factors like noise and complicated layout. Moreover, in many cases it may be difficult to evaluate OCR results against the ground truth, as reading order may be disturbed or not well defined, thereby decreasing the observed recognition quality. Finally, as both sets, cleaned and real data, refer to the same pages, it is possible to compare a recognition quality between cleaned and real life data, and hence observe how it decreases when layout gets more complicated and noise appears.

3.1.1 Ground truth preprocessing

In order to compare OCR engine output and expected ground truth, the latest had to be transformed into comparable form of a plain text file. This was achieved by XmlToTxt CLI tool:

```
The following options are required:     --xml
Usage: pl.psnc.synat.a12.aletheia.XmlToTxt [options]
  Options:
        --help          print this help
        --tabu          file name of a text file containing list of noise word's
                        ids
        --tabu-types    list of TextRegion types to omit during processing
                        Default: []
  *     --xml           file name of a page xml input
```

that reads given XML file and outputs plain text to the application's standard output. With tabu and tabu-types switches, one can filter out noise words and text regions by passing their id or type respectively. Please refer to Section 2.1 for more details regarding ground truth and text filtering.

Conversion tool was used for two groups of test pages. In case of cleaned pages, filtering options were set conforming to the cleaning process. In case of real pages, no filtering was applied.

Generated plain text is ordered according to reading order defined in the ground truth. However, IMPACT ground truth does not define complete order for all text regions (e.g. for page numbers

or marginalia). In such cases, XmlToTxt tool generates text according to internally defined rules. The text regions ordering issue is important as the verification step compares OCR engine output and ground truth basing on the plain text. As both OCR engines and ground truth transforming tool may differently align parallel text regions, the overall recognition quality score is biased. This issue was eliminated in case of cleaned pages dataset, that contain simplified layout and hence well defined reading order, but appears in case of the real pages dataset.

During ground truth XML files preprocessing few problems were encountered.

1. Characters encodings used in texts differ on different levels of ground truth. For example on glyphs level, character is encoded using combining macron, whereas on the level of words, the same character is encoded as a single unicode point. Moreover, in some situations, the same glyph is represented by different characters on different levels. In such cases the ground truth data was manually corrected and single unicode point was used.
2. XmlToTxt tool provides an option to exclude words from processed text basing on a provided list of word's identifiers. This requires that data is iterated on the words level, but there is no word order defined in the PAGE XML file. Currently, words are processed according to text equivalent from the text line level. However, this cannot be fully automated as the same word may appear more than once in a single text line, or even may not be found due to different characters encoding or replacements (compare with a previous issue). In such cases, manual correction must be involved.

3.1.2 Evaluation criteria

The evaluation process was performed using the IMPACT developed tools and resources. The National Centre of Scientific Research "DEMOKRITOS" (NCSR) has provided a set of tools for OCR evaluation. The criteria of the evaluation of these tools were described in details in one of the IMPACT project deliverables: D-OC3.1 EVALUATION METRICS AND SCENARIOS.

In scope of this report the investigated criteria was OCR accuracy both on character and word level. The calculation of the OCR accuracy was performed using the evaluation tools provided by the NCSR IMPACT project partner. The following command has been executed for each of the file under evaluation tests:

```
OCREval.exe a - IN_GT_FILE IN_OCR_RES OUT_CHAR_RES OUT_WORD_RES OUT_RES_SUMMARY, where
       IN_GT_FILE - input file - ground truth file
       IN_OCR_RES - input file - results file produced by evaluated OCR engine
       OUT_CHAR_RES - output file - results on a character level (txt file)
       OUT_WORD_RES - output file - results on a word level (txt file)
       OUT_RES_SUMMARY - output file - xml results file of OCR evaluation
```

The results of the evaluation on particular page level has been summarised in order to obtain the overall OCR accuracy on particular experiment level (e.g. for particular document where multiple pages has been evaluated). The overall accuracy for particular experiment was an effect of the following equation:

$$R_s = \frac{\sum_{1}^{n} c_i}{\sum_{1}^{n} a_i}$$

where *n* is the number of evaluated files, c_i is the number of correctly recognised characters/words in particular page and a_i is the number of all characters/words in particular page.

3.2. OCR recognition accuracy results

OCR results for FineReader are presented on Table 5, OCR results for Tesseract are presented on Table 6. Both tables have the same structure. Each row in the table represents one experiment performed on particular dataset (e.g. document) level using particular OCR engine trained with a defined number of pages. The table is composed of the following columns:

- Document - document that was the subject of the training and OCR, it refers to the dataset name mentioned in section 3.1.
- Type of document - the type of font used in the document. There are two possible values: “gothic” - for gothic (fraktur) documents and “antiqua” for antiqua documents.
- Number of trained pages - number of pages from the document that were used to train OCR engine. If the number is equal to 0 it means that no training was performed.
- Character level OCR accuracy - accuracy of the OCR engine recognition on a character level, calculated as follows:

  $$A_c = 1 - \frac{e}{c}$$

  where *e* is the number of character errors (total for insertions, substitutions and deletions), and *c* is the number of all characters in the document.

  The character level OCR accuracy is divided into two sub-columns: “cleaned” for preprocessed pages and “original” for original pages. Please refer to the section 3.1 for details on distinction between “cleaned” and “original”.
- Word level OCR accuracy - accuracy of the OCR engine recognition on a word level, calculated as follows:

  $$A_w = 1 - \frac{e}{w}$$

  where *e* is the number of word errors, and *w* is the number of all words in the document.

  The word level OCR accuracy is divided into two sub-columns: “cleaned” for preprocessed pages and “original” for original pages. Please refer to the section 3.1 for details on distinction between “cleaned” and “original”.

The FineReader and Tesseract results were generated in an incremental manner. It means that during the training the OCR process was executed after each trained page. As a result for each dataset that was processed it was possible to obtain several OCR results which correspond to

particular number of trained pages. This experiment has been performed to get an overview on how the increase of the training data influences the recognition rate of the OCR engine.

Table 5. OCR recognition accuracy results – FineReader

Document	Type of document	Number of trained pages	Character level OCR accuracy		Word level OCR accuracy	
			original	cleaned	original	cleaned
test-na	antiqua	0	81,72%	81,73%	57,98%	55,85%
test-na	antiqua	1	81,63%	83,16%	57,80%	58,27%
test-na	antiqua	15	82,89%	86,59%	60,35%	64,83%
test-na	antiqua	22	83,15%	86,63%	60,69%	64,84%
test-na	antiqua	28	83,08%	86,97%	60,42%	65,43%
test-oe	antiqua	0	61,95%	79,63%	42,46%	52,67%
test-oe	antiqua	1	68,05%	88,01%	53,89%	66,74%
test-oe	antiqua	2	68,82%	88,86%	55,26%	68,64%
test-oe	antiqua	3	69,18%	89,14%	55,54%	68,81%
test-oe	antiqua	4	69,57%	89,47%	56,28%	69,26%
test-oe	antiqua	5	69,40%	89,69%	55,83%	69,92%
test-zr	antiqua	0	75,65%	85,91%	67,11%	66,91%
test-zr	antiqua	1	78,91%	89,07%	74,01%	73,46%
test-zr	antiqua	2	79,00%	89,52%	74,30%	74,49%
test-zr	antiqua	3	79,30%	89,86%	74,99%	75,45%
test-zr	antiqua	4	79,68%	90,13%	76,17%	76,23%
test-zr	antiqua	5	79,90%	90,32%	76,41%	76,70%
test-zr	antiqua	6	79,97%	90,46%	76,53%	77,07%
test-zr	antiqua	7	80,15%	90,90%	77,31%	78,05%
test-14	gothic	0	47,86%	48,79%	14,63%	15,50%
test-14	gothic	4	82,37%	84,96%	54,54%	58,22%
test-16	gothic	0	48,04%	48,93%	14,21%	14,77%
test-16	gothic	5	73,25%	81,86%	40,11%	51,74%
test-19	gothic	0	43,26%	38,77%	11,50%	9,58%
test-19	gothic	1	52,07%	47,91%	19,12%	17,85%
test-19	gothic	2	52,65%	50,90%	21,01%	21,64%
test-19	gothic	3	52,90%	73,96%	20,82%	37,20%
test-19	gothic	4	52,79%	73,98%	20,74%	36,99%
test-21	gothic	0	51,78%	52,84%	12,70%	13,05%
test-21	gothic	1	78,22%	81,26%	41,96%	45,50%
test-21	gothic	2	79,59%	82,78%	44,71%	47,84%
test-21	gothic	3	80,11%	83,31%	43,80%	48,98%
test-21	gothic	4	80,48%	83,78%	44,42%	49,99%
test-22	gothic	0	49,85%	54,15%	17,89%	19,56%

test-22	gothic	1	59,94%	62,40%	28,84%	31,73%
test-22	gothic	2	62,16%	64,79%	30,56%	33,88%
test-22	gothic	3	62,51%	65,02%	31,48%	34,26%
test-22	gothic	4	74,87%	79,23%	48,90%	55,98%
test-22	gothic	5	74,61%	79,17%	48,61%	55,79%
test-22	gothic	6	77,28%	81,41%	52,82%	59,65%
test-22	gothic	7	78,18%	82,03%	54,47%	61,10%
test-22	gothic	8	79,32%	82,26%	57,11%	61,74%
test-22	gothic	9	79,23%	82,51%	56,86%	62,15%
test-3	gothic	0	53,91%	54,77%	16,45%	17,16%
test-3	gothic	1	78,77%	79,87%	46,70%	48,20%
test-3	gothic	2	81,77%	82,71%	51,21%	53,11%
test-3	gothic	3	82,87%	83,68%	53,30%	55,01%
test-3	gothic	4	83,34%	84,41%	53,57%	56,08%
test-3	gothic	5	83,70%	85,10%	54,31%	57,40%
test-3	gothic	6	84,01%	85,11%	54,95%	57,17%

The results for FineReader are depicted on four charts below. Each chart has on its Y axis the recognition rate expressed in %. X axis represents the number of trained pages. Chart 1 and chart 2 represent results for the antiqua documents and chart 3 and chart 4 for the gothic documents. All these charts represent tests performed on original types of pages.

Chart 1. Character level OCR accuracy in the context of the training data size (antiqua documents)

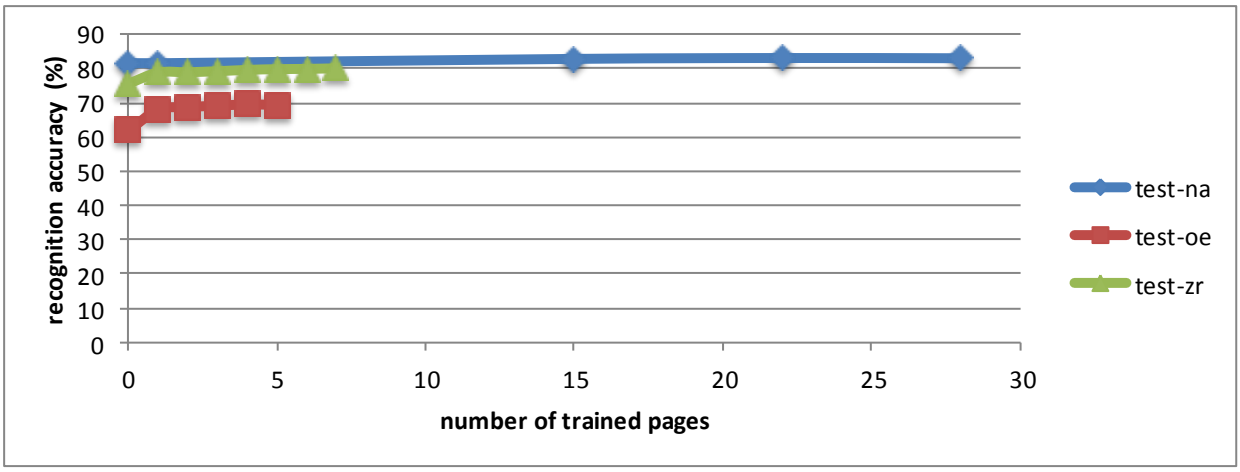


Chart 2. Word level OCR accuracy in the context of the training data size (antiqua documents)

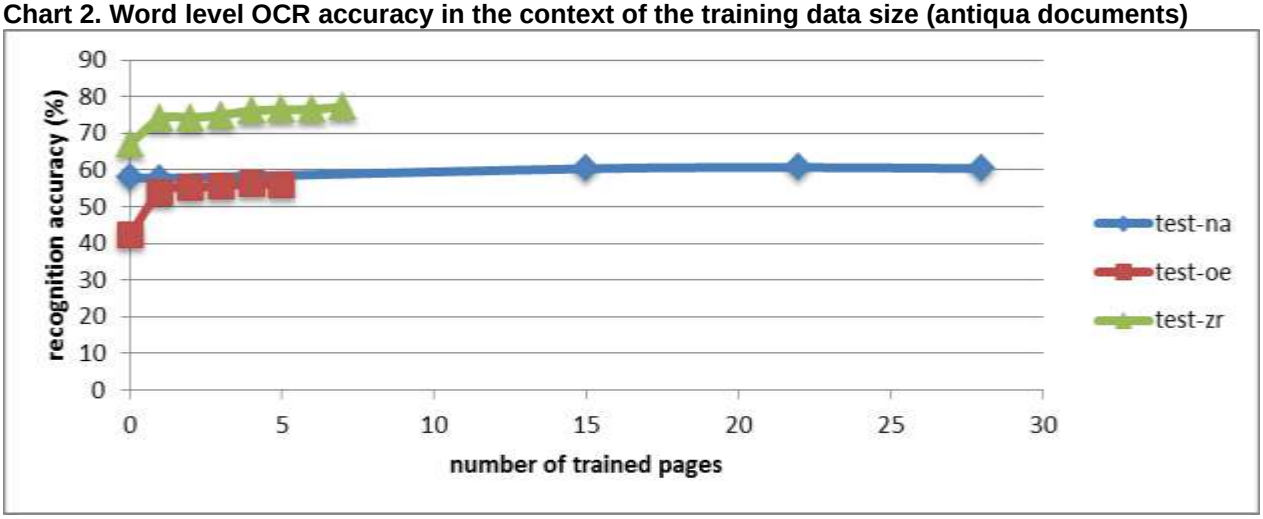


Chart 3. Character level OCR accuracy in the context of the training data size (gothic documents)

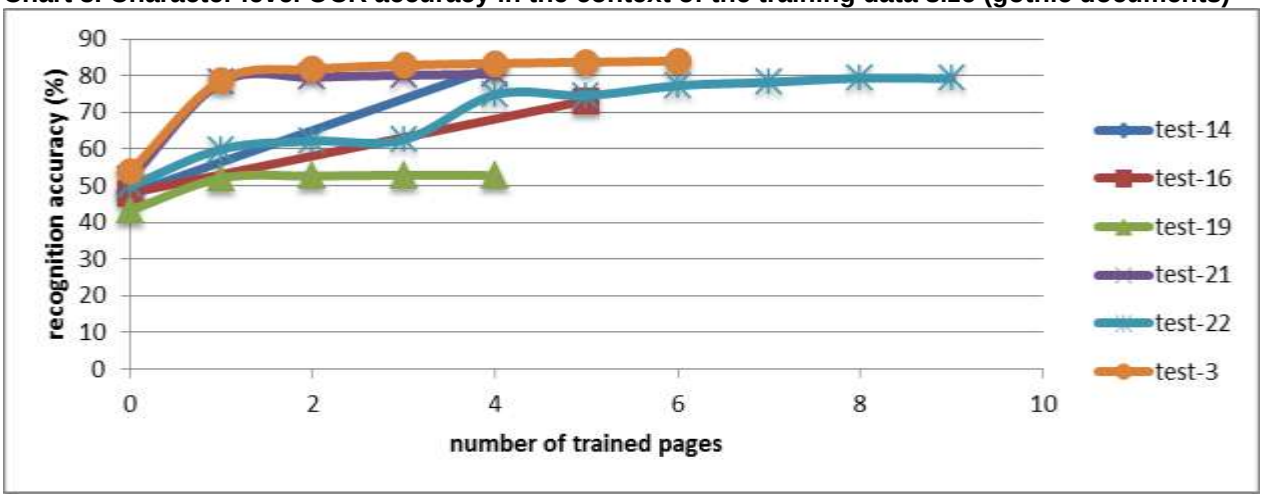
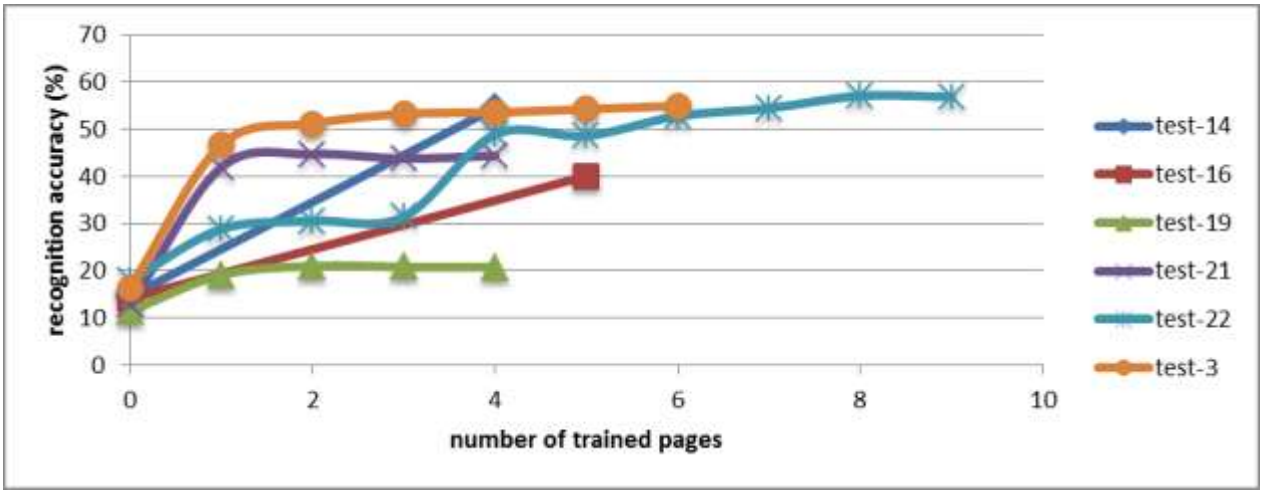


Chart 4. Word level OCR accuracy in the context of the training data size (gothic documents)



In general all the results indicate that with the increase of the training data the OCR results are getting better. Results for particular documents show that after the second trained page the OCR results increases considerably. Additional pages increase the results on a smaller scale, and can even slightly decrease the recognition rate. The decrease is probably an effect of noise introduction (too many representations of particular character). The recognition rate for the antiqua documents is approx. 80% on character level and more than 60% on word level. Although the results are not very good it proves that with the use of small number of training data the OCR results can be considerably improved and used at least for the purposes of search. The “test-oe” has given very bad results due to the bad quality images and physical degradation of the original document. On the other hand “test-zr” has given a very good results, because after only two pages being trained it was possible to reach almost 80% of character recognition and over 70% of word recognition rate.

Although the recognition rate for the antiqua documents has been considerably increased after the training process, the recognition rate of the gothic documents has been improved even more. The initial recognition rate (no training) of the gothic documents oscillates around 40-50% for the character level and 10-20% for the word level. After the training process the recognition rate has been dramatically improved and reaches 80% for the character level and 60% for the word level recognition in the best case scenarios.

Table 6. OCR recognition accuracy results – Tesseract

Document	Type of document	Number of trained pages	Character level OCR accuracy		Word level OCR accuracy	
			original	cleaned	original	cleaned
test-all	antiqua	38	49,23%	76,06%	29,77%	43,10%
test-na	antiqua	1	72,01%	83,56%	44,99%	57,42%
test-na	antiqua	15	62,92%	80,94%	35,90%	47,07%
test-na	antiqua	22	64,69%	81,91%	39,52%	51,51%
test-na	antiqua	28	69,39%	84,82%	42,52%	55,00%
test-oe	antiqua	1	37,09%	72,41%	39,61%	57,40%
test-oe	antiqua	2	40,24%	74,35%	42,29%	60,64%
test-oe	antiqua	3	38,66%	74,86%	40,21%	61,19%
test-oe	antiqua	4	36,36%	74,39%	39,38%	59,82%
test-oe	antiqua	5	75,21%	86,52%	61,19%	70,28%
test-zr	antiqua	1	47,77%	82,31%	52,93%	61,09%
test-zr	antiqua	2	50,20%	83,82%	56,62%	64,69%
test-zr	antiqua	3	48,74%	84,46%	56,87%	63,96%
test-zr	antiqua	4	48,96%	85,23%	60,16%	68,33%
test-zr	antiqua	5	46,86%	82,27%	51,63%	58,92%
test-zr	antiqua	6	47,78%	80,94%	56,52%	64,85%
test-zr	antiqua	7	39,38%	78,32%	52,72%	60,67%
test-14	gothic	4	70,62%	85,95%	49,88%	59,78%
test-16	gothic	5	66,65%	87,26%	42,19%	60,22%

test-19	gothic	1	62,73%	73,13%	34,66%	36,68%
test-19	gothic	2	65,72%	75,59%	37,45%	40,11%
test-19	gothic	3	70,26%	79,87%	40,13%	44,46%
test-19	gothic	4	71,00%	80,65%	41,17%	45,63%
test-21	gothic	1	84,27%	92,08%	59,44%	67,83%
test-21	gothic	2	84,92%	92,22%	57,98%	65,99%
test-21	gothic	3	81,80%	89,33%	53,91%	61,06%
test-21	gothic	4	83,91%	91,71%	59,01%	66,00%
test-22	gothic	1	34,99%	55,53%	12,19%	19,67%
test-22	gothic	2	59,59%	77,97%	32,34%	49,78%
test-22	gothic	3	63,28%	80,75%	40,58%	58,27%
test-22	gothic	4	64,76%	81,50%	43,20%	60,04%
test-22	gothic	5	63,23%	80,15%	37,28%	51,86%
test-22	gothic	6	64,99%	82,35%	44,54%	63,10%
test-22	gothic	7	60,26%	78,71%	37,35%	53,92%
test-22	gothic	8	57,28%	77,88%	33,99%	51,83%
test-22	gothic	9	60,17%	86,94%	39,01%	63,24%
test-3	gothic	1	79,43%	86,62%	53,67%	59,12%
test-3	gothic	2	79,76%	86,80%	54,40%	59,57%
test-3	gothic	3	77,55%	84,46%	45,17%	49,84%
test-3	gothic	4	77,09%	84,21%	48,52%	53,53%
test-3	gothic	5	77,86%	85,68%	50,78%	57,42%
test-3	gothic	6	77,30%	86,13%	50,24%	57,54%
test-all	gothic	59	54,27%	72,43%	30,77%	39,24%

Tesseract results have been calculated not only on particular document level, but also on a level of all documents of particular type. It means that during the test phase it was possible to obtain the results for the overall recognition rate on antiqua documents and gothic documents separately. The results (test-all for gothic and antiqua) indicate that the overall training process for Tesseract OCR engine has not been successful. The recognition rate of approx. 50% on character level and approx. 30% on word level cannot be considered as valuable resource. The reason is most probably related to the various document layout, font type and noise. These characteristics highly influence the Tesseract recognition rate which results in poor quality output.

On the other hand the recognition rate on particular document level is promising, as it oscillates around 70% on character level and 50% on word level. As mentioned before, the crucial element which highly decreases the recognition rate is poor layout analysis and noise reduction of the Tesseract engine.

Incremental training results for Tesseract are depicted on four charts below. Each chart has on its Y axis the recognition rate expressed in %. X axis represents the number of trained pages. Chart 5 and chart 6 represent results for the antiqua documents and chart 7 and chart 8 for the gothic documents. All these charts represent tests performed on original types of pages.

Chart 5. Character level OCR accuracy in the context of the training data size (antiqua documents)

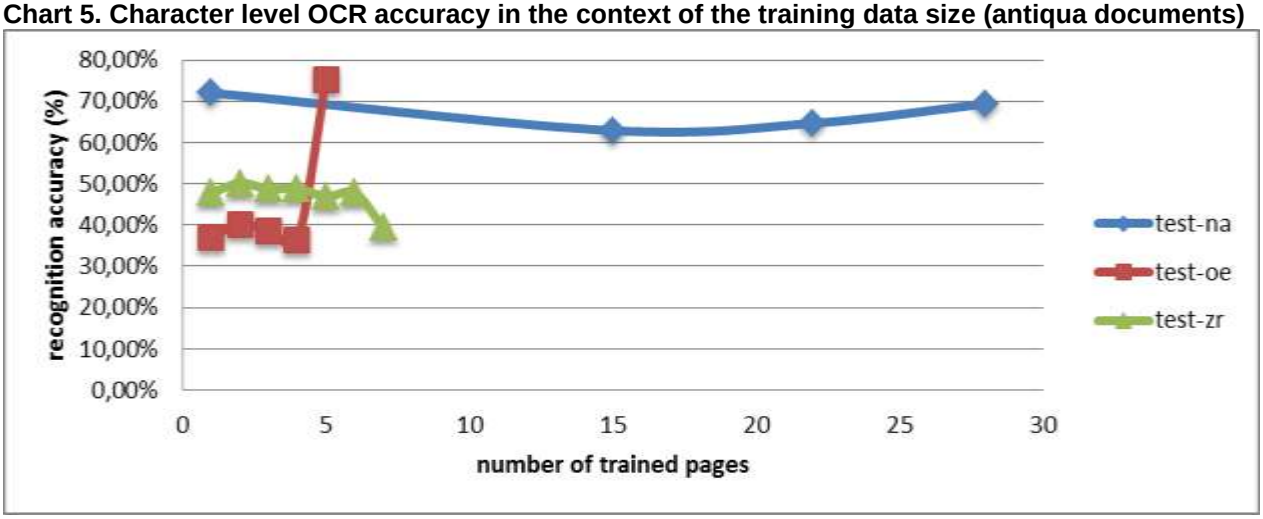


Chart 6. Word level OCR accuracy in the context of the training data size (antiqua documents)

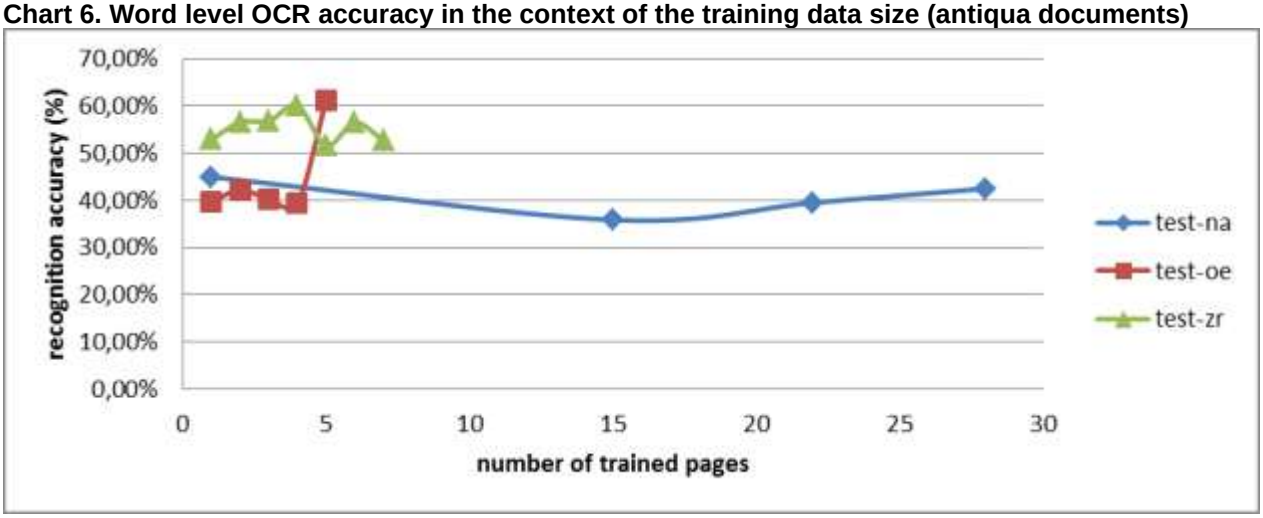


Chart 7. Character level OCR accuracy in the context of the training data size (gothic documents)

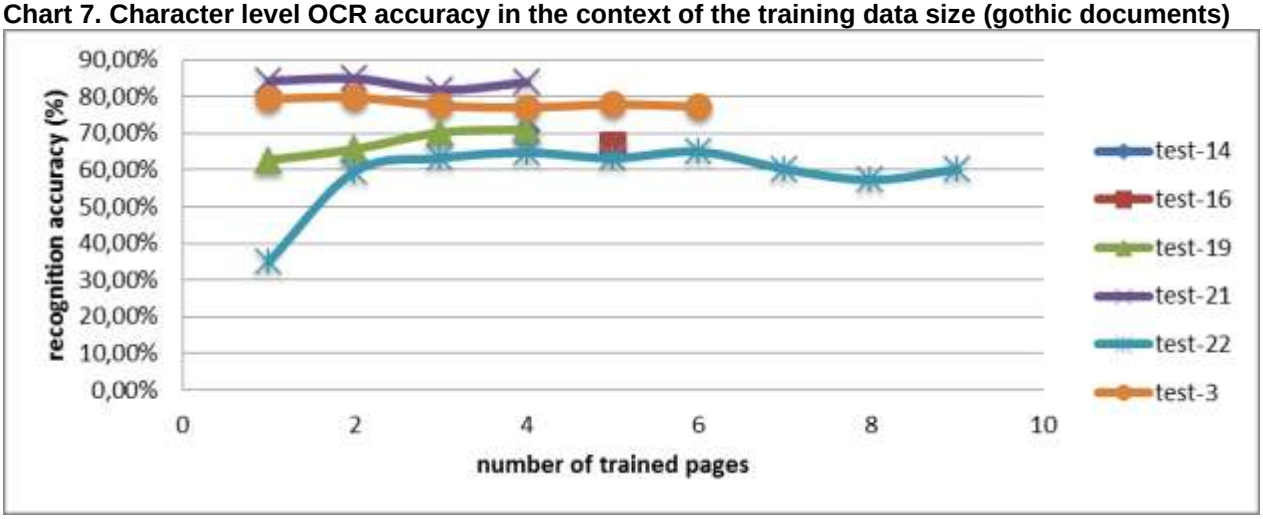
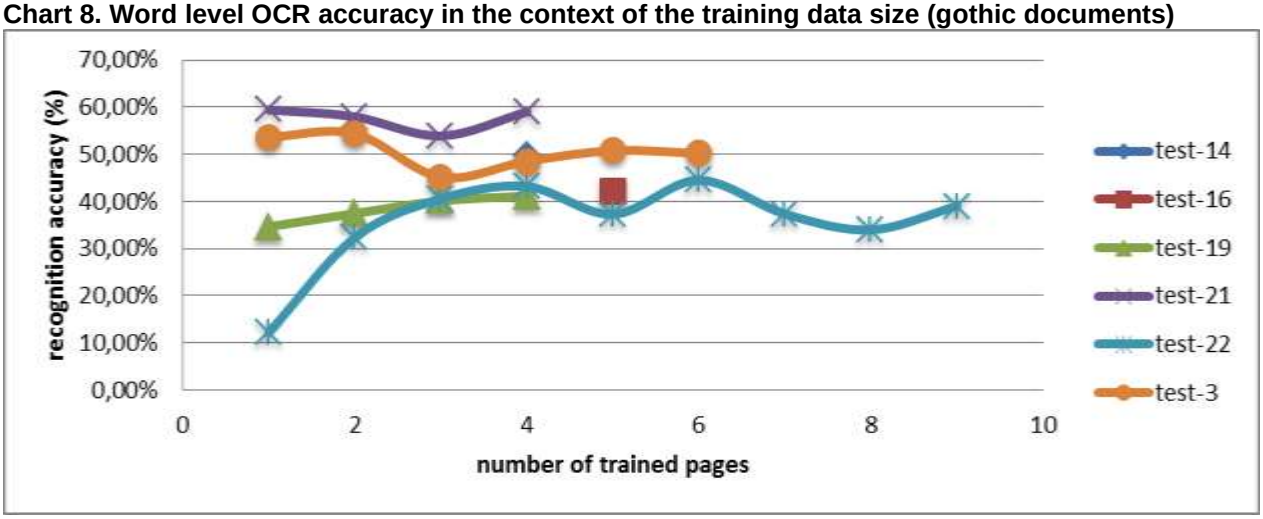


Chart 8. Word level OCR accuracy in the context of the training data size (gothic documents)



The incremental results for Tesseract show, that although usually the subsequent trained pages increase the accuracy, they can also introduce noise. In most cases after approximately 3rd trained page the accuracy decreases (noise introduction). Then the remaining pages increase the accuracy, but the accuracy does not get much better than after 2nd trained page or can even remain worse. This is most probably due to the poor Tesseract noise handling and layout analysis. With the growth of training patterns Tesseract starts to recognise noise as false positives characters. So with more different patterns coming from more pages, Tesseract recognises more false positives and reduces the accuracy (e.g. in case of test-zr).

3.3 Comparison of Tesseract and FineReader recognition accuracy

In order to compare the Tesseract and FineReader OCR engines several tests has been performed. All these tests has been individually analysed in the scope of particular OCR engine. The results of particular tests with all the details related to these results has been described in section 3.2. For the purposes of the comparison the overall results on particular document level has been presented on table 7. Each row in the table represents one experiment performed on particular document level using both FineReader and Tesseract OCR engines, trained with the same pages. The table is composed of the following columns:

- Document - document that was the subject of the training and OCR, it refers to the dataset name mentioned in section 3.1.
- Type of document - the type of font used in the document. There are two possible values: “gothic” - for gothic (fraktur) documents and “antiqua” for antiqua documents.
- Type of pages - the type of pages that were OCRed. Two options are possible: “cleaned” and “original”. Please refer to the section 3.1 for details on distinction between these two possible values.
- Number of trained pages - number of pages from the document that were used to train OCR engines.
- Character level OCR accuracy - accuracy of the OCR engine recognition on a character level, calculated as follows:

$$A_c = 1 - \frac{e}{c}$$

where *e* is the number of character errors (total for insertions, substitutions and deletions), and *c* is the number of all characters in the document.

The character level OCR accuracy is divided into two sub-columns: FineReader for recognition rate of FineReader OCR engine and Tesseract for recognition rate of Tesseract OCR engine.

- Word level OCR accuracy - accuracy of the OCR engine recognition on a word level, calculated as follows:

$$A_w = 1 - \frac{e}{w}$$

where *e* is the number of word errors, and *w* is the number of all words in the document.

The word level OCR accuracy is divided into two sub-columns: FineReader for recognition rate of FineReader OCR engine and Tesseract for recognition rate of Tesseract OCR engine.

Table 7 Comparison of OCR accuracy results: FineReader vs Tesseract

Document	Type of document	Type of pages	Number of trained pages	Character level OCR accuracy		Word level OCR accuracy	
				FineReader	Tesseract	FineReader	Tesseract
test-na	antiqua	cleaned	28	86,97139	84,81774	65,43104	54,99568
test-na	antiqua	original	28	83,08036	69,38797	60,420418	42,524837
test-oe	antiqua	cleaned	5	89,68817	86,52127	69,91817	70,27824
test-oe	antiqua	original	5	69,399185	75,21368	55,82787	61,185528
test-zr	antiqua	cleaned	7	90,89958	78,31711	78,05078	60,667484
test-zr	antiqua	original	7	80,146996	39,377434	77,30597	52,722473
test-14	gothic	cleaned	4	84,95764	85,94833	58,21762	59,782887
test-14	gothic	original	4	82,3733	70,61988	54,54097	49,876606
test-16	gothic	cleaned	5	81,85919	87,26151	51,738323	60,224297
test-16	gothic	original	5	73,24994	66,64795	40,109287	42,185795
test-19	gothic	cleaned	4	73,9826	80,645386	36,985195	45,625843
test-19	gothic	original	4	52,791237	70,99617	20,743692	41,16866
test-21	gothic	cleaned	4	83,78464	91,71215	49,986412	65,99619
test-21	gothic	original	4	80,47636	83,9082	44,417477	59,00755
test-22	gothic	cleaned	9	82,50946	86,93861	62,151653	63,23807
test-22	gothic	original	9	79,22829	60,16857	56,864452	39,0144
test-3	gothic	cleaned	6	85,106636	86,12579	57,17473	57,535183
test-3	gothic	original	6	84,005486	77,30411	54,95192	50,24464

In general in case of gothic type documents Tesseract provides more accurate results both in terms of character and word level accuracy and both in case of “cleaned” and “original” types of pages, but the advantage is more visible in case of “cleaned” pages tests. Particular tests and comparison of the FineReader and Tesseract OCR results give an initial insight into the common problems and challenges that should be considered in future works related to both tools. The following conclusions have been derived during the analysis of the results for particular tests:

- test-na: FineReader is more accurate on characters level than Tesseract in case of both cleaned data and real data.
- test-oe: FineReader is slightly more accurate on characters level than Tesseract in case of cleaned data, whereas Tesseracts wins in the real data test case. This dataset contains images of a good quality, with a high contrast between glyphs and background.
- test-zr: FineReader is more accurate on characters level than Tesseract in case of cleaned data, and definitely more accurate in case of real data, when Tesseract didn’t correctly recognise page boundaries, generating a lot of false positive characters.
- test-14: Tesseract engine has serious problems with the layout analysis and therefore the OCR results for the original type of pages is worse than FineReader.
- test-16: Similarly to the test-14 Tesseract gave worst results in case of original pages. The reason in this case was mainly the noise on the scans, which was usually interpreted by Tesseract as additional characters, resulting in “addition” type of errors.
- test-19: FineReader gives very bad results – approx. 45% on character level – for pages with greater amount of characters (small characters, approx. 2500 characters per page), it gives much better results for pages with less characters (larger characters, approx. 1600 characters per page) – the accuracy for such cases is approx. 76% on character level. It is important to note that the engine was trained to recognise both small characters and larger ones.
- test-21: Good quality images give the Tesseract considerable advantage over the FineReader. The Tesseract is better both in the “cleaned” and “original” test.
- test-22: The complicated layout (e.g. mariginalia) and visible noise causes Tesseract to loose in the “original” test.
- test-3: Similarly to the test-22 the complicated layout (e.g. mariginalia) and visible noise causes Tesseract to loose in the “original” test.

Finally, the facts related to the efforts needed to train particular OCR engine are described in this paragraph. This cannot be considered as a direct difference or comparison on the financial requirements related to training particular engine, nevertheless it can give an impression on how certain approaches involve available personnel and what are the approximate estimates for getting usable OCR results, both for Tesseract and FineReader. In case of Tesseract PSNC staff (IT specialists) needed approximately 15 working days to prepare all the necessary tools required for Tesseract training. It is important to note that the tools can be further used for any number of training experiments. Additional 8 days were required to train Tesseract in scope of the datasets investigated in this pilot work with the use of the developed tools. In case of FineReader it was required approximately 25 working days to train it both on gothic and antiqua documents. FineReader training did not include the test-all cases for gothic and antiqua tests. Assuming that the tools developed for Tesseract are available, the staff required to train Tesseract and FineReader is similar. It means that it is required to have a person familiarised with appropriate software tools (developed by PSNC for Tesseract or FineReader itself) and educated in the context of the documents being processed, e.g. for historical documents a specialist is required in order to identify all the characters available on the scanned pages. Moreover the Tesseract training required ground truth data while FineReader did not. This is

because the of the training approach: automatic for Tesseract and manual for FineReader. It is also clearly visible in terms of the efforts needed to train certain OCR engine: 8 days for Tesseract and 25 days for FineReader. In order to obtain ground truth data for Tesseract (beside the test-all cases) it was required to invest approximately 700 EUR (subcontract). Although the Tesseract training related to whole antiqua and gothic test did not give satisfactory results, it can be interesting to mention that ABBYY has offered to train FineReader on Polish gothic documents for the price between 31 700$ and 45 000$, but no estimates on the resulting accuracy could be given (the estimate given in February 2012). It shows that the effort needed to train OCR engine for a general dataset, including various types of documents is probably time-consuming task and requires considerable effort.

4 Conclusions

When comparing results of both engines in test, there is no single winner that would outperform the second engine in all test cases. However, we try to point out differences between FineReader and Tesseract engines.

First of all, Tesseract seems to deal better than FineReader in case of gothic tests in their cleaned form, where most of glyphs are of a fraktur font. This may be caused by the fact that Tesseract's training included only fraktur glyphs, whereas FineReader contains an embedded set of modern glyphs patterns and hence the glyph's classification is more error prone. Moreover, Tesseract does not handle complicated or noised page layout well. For example in case when page boundaries are represented by irregular shapes, dots and lines rather than solid areas, Tesseracts generates much false positive characters decreasing the results precision and hence accuracy value. This can be clearly observed by test-zr original antiqua test case where character level accuracy is 39.4%. In case of incremental training the observation shows that Tesseract's recognition accuracy can be decreased by the introduction of noise (in case of most tests 3rd page decreased the accuracy).

In case of FineReader it was usually enough to train two pages to get considerably better results, further training increases OCR results on a smaller but important scale. FineReader had difficulties with a test case where two kind of pages were OCRed: pages with large font (approx. 1600 characters per page) and with small font (approx. 2500 characters per page). The overall results were very bad, due to the fact that the pages with small font gave very bad results (45% on character level), despite the fact that pages with large font gave relatively good results (76% on character level).

Overall tests for gothic and antiqua documents for Tesseract give poor results, due to various types of documents and their fonts, layouts and evident noise appearing on the scanned pages. However, it can be observed that for good quality pages Tesseract gives considerably better results than FineReader.

The overall effort needed to train Tesseract and FineReader was different in nature, as the Tesseract approach was automatic and FineReader was manual. The testing activities required

more effort in case of FineReader, but in order to be able to train Tesseract it was required prepare additional ground-truth data for the pages used in training (e.g. by using the *Cutouts* tool or subcontracting the work).