SACJ 32(2) December 2020
Research Article

Exchanging image processing and OCR components in a Setswana digitisation pipeline

Gideon Kotzé, Friedel Wolff

Academy of African Languages and Science, University of South Africa

ABSTRACT

As more natural language processing (NLP) applications benefit from neural network based approaches, it makes sense to re-evaluate existing work in NLP. A complete pipeline for digitisation includes several components handling the material in sequence. Image processing after scanning the document has been shown to be an important factor in final quality. Here we compare two different approaches for visually enhancing documents before Optical Character Recognition (OCR), (1) a combination of ImageMagick and Unpaper and (2) OCRopus. We also compare Calamari, a new line-based OCR package using neural networks, with the well-known Tesseract 3 as the OCR component. Our evaluation on a set of Setswana documents reveals that the combination of ImageMagick/Unpaper and Calamari improves on a current baseline based on Tesseract 3 and ImageMagick/Unpaper with over 30%, achieving a mean character error rate of 1.69 across all combined test data.

Keywords: digitisation, optical character recognition, image processing, neural networks

Categories: • **Applied computing** ∼ **Optical character recognition** • **Computing methodologies** ∼ **Image processing**

Email:
Gideon Kotzé dr.gideon.kotze@gmail.com (CORRESPONDING),
Friedel Wolff friedel@translate.org.za

Article history:
Received: 10 May 2019
Accepted: 03 November 2020
Available online: 08 December 2020

1 INTRODUCTION

The work we present takes place in the context of a digitisation project of mostly older typewritten books in South African languages. The primary focus of the project is to extend text corpora for the languages involved, but such projects also bring advantages, such as electronic access to books that are out of print. Consequently, we aim for high-quality digital versions of the text suitable for the Digital Humanities, corpus linguistics and applications in natural language processing (NLP). The investigation into a digitisation pipeline is therefore to facilitate and accelerate the process leading to quality-assured digital versions of the text portions of the books involved.

Kotzé, G. and Wolff, F. (2020). Exchanging image processing and OCR components in a Setswana digitisation pipeline. *South African Computer Journal 32*(2), 218–231. https://doi.org/10.18489/sacj.v32i2.707

SACJ is a publication of the South African Institute of Computer Scientists and Information Technologists. ISSN 1015-7999 (print) ISSN 2313-7835 (online).

A typical pipeline consists of the following phases, applied in order: scanning, image processing, optical character recognition (OCR) and post-processing. Scanned images often contain imperfections such as skew, speckles and lack of contrast that can be improved through the application of image processing software with the intent of creating better input for OCR (see, for example, Gupta et al., 2007; Kotzé and Wolff, 2017; Patvardhan et al., 2013). OCR itself is a difficult problem to solve, which as a result produces mostly less-than-perfect results. Post-processing can then either attempt to directly solve OCR errors or focus on restoring the logical layout of the resulting digital text. Given a perfect output example in the form of a gold standard, digitisation output can be automatically evaluated using some sort of metric. Here, we apply the widely used *Character Error Rate* (CER) metric, explained in Section 5.3.

A number of Free and Open Source (FOSS) tools are available for building such a digitisation pipeline. With such tools, the ability to train our own OCR models enables us to achieve the best results possible. In this work, we compare the established Tesseract[1] package with the recently released Calamari[2] package. Since they are not directly equivalent in functionality, we have to compare Tesseract with a combination of Calamari and OCRopus tools (Breuel, 2008) to form a complete pipeline.

The success of neural networks in several NLP tasks in recent years, such as language modelling, parsing and machine translation, suggests that its use in Calamari might be competitive with the pattern-based approach of Tesseract version 3.

The rest of the paper is structured as follows: In Section 2, we discuss related work. The research questions are posed next, in Section 3. This is followed by an introduction to the Calamari package and how to train it (Section 4). Next, we discuss the digitisation pipeline in Section 5. We present the experimental data in Section 6, the experiments and evaluation results in Section 7, and a qualitative evaluation in Section 8. Finally, a discussion of the results and planned future work is presented in Section 9.

2 BACKGROUND

As mentioned, Tesseract (Smith, 2007) is a pattern-based OCR system. It implements a type of page analysis where page areas, lines, words and eventually characters are recognised through a series of processes that involve (1) a static classifier and (2) an adaptive classifier that is trained by the output of the former. It also involves a small measure of linguistic analysis, as well as some image processing, such as border analysis and handling noise.

Hocking and Puttkammer (2016) present Tesseract models for ten South African languages, achieving high accuracy in comparison with an English model on the same texts. It represents a first attempt at distributing OCR engines used by open-source software (Tesseract) for South African languages other than Afrikaans. We describe, in Kotzé and Wolff (2017), a complete pipeline showing a notable improvement over the Setswana model in Hocking and Puttkammer

[1] https://github.com/tesseract-ocr/
[2] https://github.com/Calamari-OCR/calamari

https://doi.org/10.18489/sacj.v32i2.707

(2016) when image processing and post-processing steps are added, resulting in texts needing less post-editing than without such pre- and post-processing. As with Hocking and Puttkammer (2016), this was achieved with freely available and off-the-shelf software.

OCRopus is a complete OCR pipeline comprised of separate commands. Consequently, it is trivial to swap out one of the commands for an alternative with similar functionality. Calamari is a line-based OCR package (Wick et al., n.d.) aimed to integrate into the OCRopus pipeline (Breuel, 2008). It uses Convolutional Neural Networks (CNNs) and bidirectional Long Short-Term Memory (LSTM) layers. The network is trained by the Connectionist Temporal Classification (CTC) algorithm (Graves et al., 2006). It uses the Tensorflow[3] framework for its neural network computations and consequently can use GPU processing for better performance, during both training and prediction.

The recently released version 4 of Tesseract implements a similar neural network approach, but the documentation warns that training can take weeks, and that training "is a daunting task" with caveats that can lead to reduced performance[4]. At the time of writing, Tesseract 4 does not support training on GPU hardware, which might be a central reason for the substantial requirement in training time compared to Calamari, which does offer that ability. The officially published models for Tesseract 4 do not include support for South African languages, with the exception of Afrikaans. These models were trained on hundreds of thousands of text lines, and with thousands of fonts. It is unclear if such a scale of training is required for reasonable results, or if rather a more modest amount is sufficient.

The authors of Calamari compared it to several FOSS OCR engines (including the until then unreleased Tesseract 4), and Calamari obtained some of the best results.

3 RESEARCH QUESTIONS

We have shown positive results (Kotzé & Wolff, 2017) for integrating Tesseract-based Setswana OCR into a pipeline that includes pre- and post-processing components. The fact that NLP solutions based on neural networks have made good progress in recent years, as well as the positive results reported by Wick et al. (n.d.), lead to the reasonable hypothesis that adapting such a pipeline to accommodate Calamari might lead to improved results. The question of whether or not a given mode of image processing will have the same impact on a certain OCR approach than another is also unanswered. Since Calamari has been developed to be used with the OCRopus tools, which includes its own image processing tool, we are able to combine both different image processing as well as OCR approaches within our complete pipeline and contribute towards answering this question.

With this paper, we thus attend to the following research questions: Given a selection of documents with gold standards:

[3] https://www.tensorflow.org/
[4] https://tesseract-ocr.github.io/tessdoc/TrainingTesseract-4.00

https://doi.org/10.18489/sacj.v32i2.707

1. Does the application of Calamari, everything else being equal, outperform the Tesseract model in Kotzé and Wolff (2017), and if so, to which degree?

2. Does the application of a certain combination of components in the pipeline outperform the combination of components reported in Kotzé and Wolff (2017), and if so, to which degree?

3. Can we get a measure of how generalisable our approaches are, and therefore make an informed decision on how to digitise new documents leading to less post-editing?

Here, "gold standard" refers to a plain text file against which the pipeline output is automatically evaluated, and "outperform" refers to an improvement in the CER (see Section 7).

In the following section, we discuss the Calamari package and how to train it.

4 TRAINING CALAMARI

Only a few pretrained models are available for Calamari. However, training a new model is relatively easy. To make results more comparable to the Tesseract model, we generate training lines from the Setswana portion of the NCHLT text corpus[5] comprising just over 20,000 lines and 302,508 words. The text was not normalised (or regularised in Calamari terminology) so that a wider repertoire of characters would be supported. For example, our requirement is for eventual post-edited documents to contain the full variety of typographic punctuation marks (such as "curly" typographic quotes and dashes in addition to hyphens), therefore an ideal pipeline would generate them as input to the post-editing process. For this purpose, most straight quotation marks in the text were replaced automatically with their typographical variants to ensure that they are present in the training data.

An additional 4,551 lines with almost 70,000 words served as validation set, which Calamari uses to stop training. An OCRopus script, *ocropus-linegen*, is used to generate the graphical training and validation data automatically based on a number of common open-source font files including all their varieties (such as bold, italic, etc.) comprising a total of 123 visual variants. Default training parameters were used, except for a batch size of 32 and the use of "simple" text regularisation. Training was accelerated with an NVIDIA Tesla K80 GPGPU. The training was stopped automatically after about 19 hours when performance on the validation set converged.

Training was performed with a version of Calamari directly from version control[6] after version 0.2.0 which has support for Tensorflow 1.11.

[5] https://repo.sadilar.org/handle/20.500.12185/343
[6] https://github.com/Calamari-OCR/calamari/tree/effde8d7

https://doi.org/10.18489/sacj.v32i2.707

5 PIPELINE

The processing pipeline contains the following main components:

1. Image processing improves visual properties such as contrast, removes visual noise, and deskews the scanned images so that text lines are perfectly horizontal.
2. Page and line segmentation and OCR involve the visual analysis from page level down to line level, and provide textual output.
3. Post-processing corrects mismatches in text representation between the document and the expected format, such as the use of newlines or hyphenation.
4. Evaluation compares the final output with the gold standard and summarises the discrepancies.

5.1 Image processing

Documents are scanned using our local Xerox machine. Scans are in high quality (600 DPI) and colour, in order to keep as much of the image data intact—also for archival purposes. When required, image quality can then always be downgraded and/or converted to grayscale or monochrome. Indeed, the Tesseract documentation recommends 300 DPI as input and therefore, for our experiments, scanned documents were converted to 300 DPI before further processing. The image processing software that we describe here converts input files to monochrome.

Scanning data is received in the form of PDF/A files that can either be directly OCR'd or handled by image processing software. These files are split into separate pages in order to support parallel processing. For the purpose of our experiments, we compare two alternative image processing solutions:

1. a combination of *ImageMagick*[7] and the tool *Unpaper*[8] (henceforth referred to as IMU).
2. the OCRopus script for image binarisation called *ocropus-nlbin* (henceforth referred to as *nlbin*).

ImageMagick is mainly used for removing noise, to set contrast and brightness levels, and to convert between image file formats. Unpaper performs page deskewing, rotation, as well as border and margin operations. *nlbin* performs more or less the same set of tasks as the above.

[7] https://www.imagemagick.org/
[8] https://github.com/Flameeyes/unpaper

https://doi.org/10.18489/sacj.v32i2.707

5.2 Optical character recognition

The output of image processing can be used for OCR, using either Tesseract or Calamari. This is an involved step with architectural differences between the tools we are comparing. Tesseract handles the whole process with one tool, whereas Calamari is combined with tools from the OCRopus project. As such, we are not comparing Tesseract with Calamari alone, but with Calamari combined with a few OCRopus tools that perform equivalent functions to the whole of Tesseract. We therefore use Tesseract in its usual configuration that performs some image processing and all of the page analysis in a single software component. By contrast, in the case of Calamari, the OCR step is broken down into separate page and line segmentation steps (performed by *ocropus-gpageseg*) and the OCR performed by Calamari.

For Tesseract, we have so far applied Tesseract 3 in order to be able to use the model described by Hocking and Puttkammer (2016). By default, it generates plain text, although in the pipeline, we have set its output to produce PDF in order for post-processing to take place.

5.3 Post-processing and evaluation

After applying OCR, post-processing steps can be taken in order to ensure correct logical flow of the text. For example, a set of single lines should be displayed as a paragraph instead, headers and footers could be removed in order to ensure continuity of adjoining paragraphs, and end-of-line hyphens occurring in the scan could be removed to correctly join and display the words that they are hyphenating.

Our digitisation pipeline supports the analysis of OCR output in the form of readable PDF files. Here, we only experiment with the PDF analysis tool Ontrafel, as described in Kotzé and Wolff (2017). Ontrafel's main features are the ability to use the information present in a readable PDF to reconstruct paragraphs from separate lines, recognise paragraph boundaries, isolate headings from adjoining paragraphs, normalise end-of-line hyphenation, as well as the removal of headers, footers and page numbers. Typical output is in the form of a single text file that can be evaluated against a gold standard.

As in previous work (Hocking & Puttkammer, 2016; Kotzé & Wolff, 2017), we evaluate the results using the CER metric that calculates the number of character insertions (i), substitutions (s) and deletions (d) that are needed to transform a given output into the reference text—this is then divided by the total number of characters in the reference.

$$CER = (i + s + d)/n \tag{1}$$

In this paper we express CER as a percentage, where lower scores are better. For the calculation of the CER we use the *ISRI Analytic Tools for OCR Evaluation* (Rice, 1996)[9] that do not only calculate the CER, but also provide statistics that are useful for qualitative evaluation. For this particular task (Section 8), we mainly consult, for each experiment, a generated frequency list of incorrectly recognised strings, as well as a list of all characters showing to which degree they

[9] https://github.com/eddieantonio/isri-ocr-evaluation-tools

Table 1: List of books used for experiments. We will refer to the books in the rest of the paper using the abbreviated names in italics.

Book	Author	Year
Maungo a Matsapa (*maungo*)	HJK Rathabe	1992
Go sa Baori (*baori*)	DP Semakaleng Monyaise	1994
Kgosi Isang Pilane (*pilane*)	MOM Seboni	1961
Botshelo jwa ga Jean Calvin (*botshelo*)	CP Senyatsi	unknown
Kgosi Sebele II (*sebele*)	MOM Seboni	1956
Ke ne ka rata Kgarejwana (*ke ne ka*)	Walter Trobisch	1966
Ke rata Lesogana (*lesogana*)	Walter Trobisch	1967
Go Ipaakanyetsa Nyalo (*nyalo*)	J Marshall	1974
Rammônê Wakgalagadi (*rammônê*)	MOM Seboni	1947

Table 2: Descriptive statistics on the documents used for experiments. The footnotes in *lesogana* have been deleted for the sake of continuous flow of paragraphs, leading to a slightly worse score.

Book	Features	Pages / Total	Words
maungo	few marks	62 / 64	16,442
baori	occasional small margins, no marks	123 / 126	60,676
pilane	occasional small margins; few pen marks	55 / 60	24,481
botshelo	5 pictures; some bleedthrough	65 / 68	14,821
sebele	occasional small margins; some pen underlined text	117 / 124	36,891
ke ne ka	no marks	33 / 36	14,415
lesogana	moderate amount of pencil marks, two footnotes	60 / 62	26,413
nyalo	4 pictures, 6 diagrams; some underlined text; typed	166 / 178	40,331
rammônê	moderate pencil marks, some pages with verse form	98 / 102	27,652

were correctly recognised in the text. This information can quickly show us if, for example, line segmentation can go awry (long strings that are missing in the output), specific characters are especially problematic for OCR, or to gauge the contribution of post-processing (better handling of newline characters, for example).

Finally, the option remains to normalise a given text before evaluation occurs. We have mainly applied normalisation to single and double quotation marks—this is discussed further in Section 7.

https://doi.org/10.18489/sacj.v32i2.707

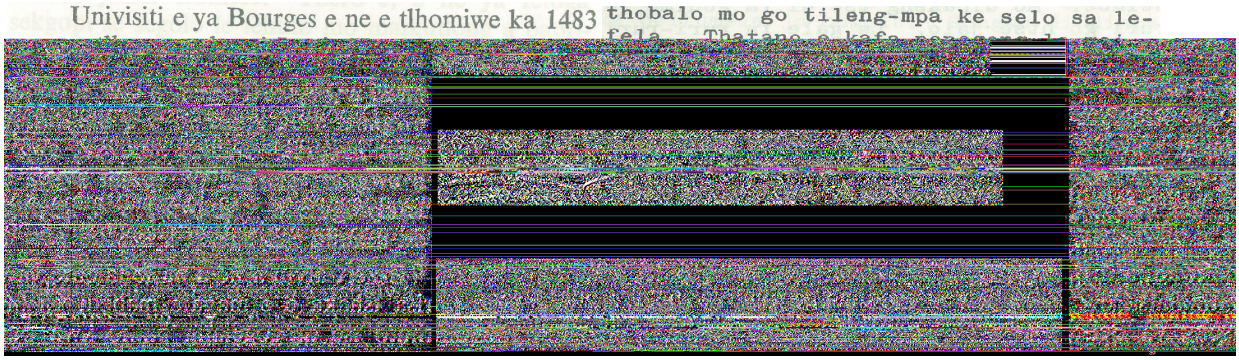

Figure 1: From left to right, clockwise: (1) Bleedthrough in *botshelo* (2) Typewriter font in *nyalo* with underlined words. Also note the Afrikaans word "sielkundige" (psychologist) that may present additional problems for OCR. (3) Underlining with pencil in *rammônê*. (4) Underlining with pen in *sebele*.

6 DATA

In our experiments (Section 7) we used a selection of nine Setswana books from the Unisa library on its main campus, written between 1947 and 1994[10] in a variety of fonts, and containing a mix of fiction and non-fiction. Table 1 contains more information. Note here that for each book, we have added an abbreviated name that we will use to refer to it from now on. The same names are used in Tables 2, 3, 4 and 5, as well as in Figures 1 and 2. All of these works are copyrighted and hence, our dataset is currently not available for distribution.

The books have been scanned, digitised (using the aforementioned Tesseract model) and quality assured by a mother-tongue expert, mainly for the purposes of inclusion into a text corpus. However, we found that they are prime candidates for evaluating digitisation as well: the books are mostly of good quality, they contain few pictures and the text layout is very simple, with few to no complicating elements such as footnotes, separate text blocks or double columns.

Nevertheless, we decided to pre-process the books by removing a selection of pages. These mainly include any title pages, tables of contents, pages consisting of only graphics, and pages with irregular layouts such as those often found in the back section. Although we digitise these pages for our corpus as well, we argue that for the purpose of testing a digitisation approach, correct OCR recognition of the exact number of leader dots in the table of contents or text in an irregular font printed on top of a graphic, for example, is of little scientific interest. Furthermore, as the largest part of our data consists of simple, flowing text, we are more interested in the performance of the digitisation pipeline on this particular aspect.

In *botshelo*, four pages contain both graphics and text (excluding captions), whereas in *nyalo*, six pages contain text pages with diagrams. Similarly, the first page of *maungo* contains a stamp and some stickers added by the library. As more text is better, and to obtain a measure of how well the systems can handle the distinction, we decided to leave these pages in.

[10] One of them has no discernible date.

https://doi.org/10.18489/sacj.v32i2.707

Table 2 presents some descriptive statistics on the texts, while Figure 1 contains some examples of notable characteristics described in the table. The total word count of all used pages is 262,122.

7 EXPERIMENTS AND EVALUATION RESULTS

Since we are comparing complete pipelines on a full document digitisation task, we can only report on the results of the whole pipeline, and not of the OCR alone. That, however, provides an evaluation of the choices that are currently easily available—not all components can be mixed and matched at will.

We report using a combination of the Setswana and English models of Tesseract (Kotzé & Wolff, 2017). In order to keep the results more comparable, we use the Setswana model only, since the Calamari model was not trained on English data.

In preparatory work, we found that setting the parameter values of ImageMagick's brightness and contrast settings—to -20 and 80, respectively—improved the visual attributes of the scanned pages, since the default values did not result in usable output. The same parameter values were used for all books, with the exception of *rammônê*, where our default setup led to many blank spots on the pages. In this case, applying the default values of the black and white threshold parameters for Unpaper led to much more acceptable results. Note that in order to avoid bias, we did not choose values based on evaluation output scores, and hence, this process requires manual inspection for each book used in our experiments. In the case of *nlbin*, as well as with Tesseract's image processing, only default values were used.

An obvious shortcoming of the Tesseract model that emerged from evaluation is the lack of typographically correct quotation marks. In a project where such marks are required, this shortcoming can be addressed reasonably successfully by post-processing—a simple script can substitute straight quotes with their typographic alternatives based on a simple consideration of context. Although this approach is not perfect (e.g. an apostrophe at the start of a word is bound to rather be converted to an opening single quote), it provides an easy way to compensate for this shortcoming of the Tesseract pipeline. Typographic quotation marks were used in all books except in *nyalo* that was prepared with a typewriter.

Since some straight quotation marks could still be generated with the Calamari pipelines, we perform the post-processing for all systems.

The additional post-processing to correct for typographic quotes resulted in a marked improvement for two books (*maungo* and *baori*) when processed with the Tesseract pipelines, and slight improvements for all the others, apart from *nyalo* that was not post-processed (see Figure 2 and table 3). The following can however be noted about Calamari with IMU: It has the best average (1.69) and median (1.07) scores, and additionally its minimum (0.24) and maximum (4.97) scores are the lowest. The standard deviation is also slightly lower, which suggests more consistent performance. If *nyalo* is removed from the evaluation, however, Tesseract with *nlbin* has the best average, standard deviation and maximum.

https://doi.org/10.18489/sacj.v32i2.707

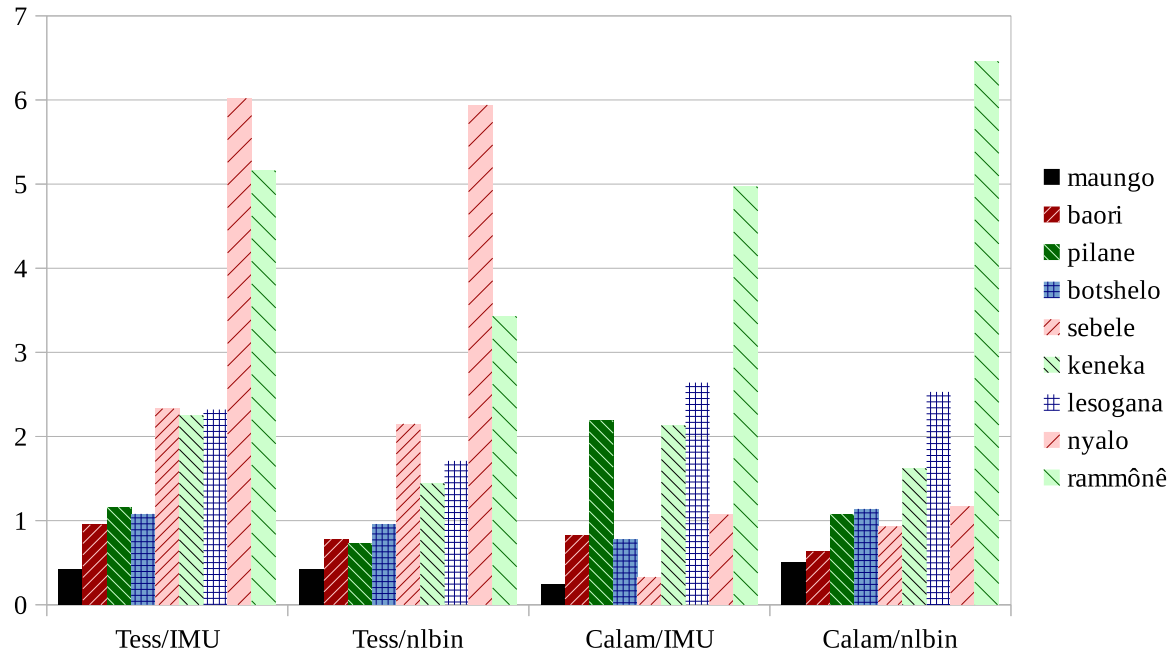

Figure 2: CER for all four systems on post-processed data

Tesseract seems to perform best with *nlbin* rather than IMU, whereas Calamari with IMU usually outperforms Calamari/*nlbin*—despite the fact that Calamari's own documentation recommends using it with the OCRopus tools. The largest difference between Tesseract and Calamari is with *nyalo*. This book was prepared on a typewriter, and the lower-case letter 'l' is visually indistinguishable from the digit '1'. Here, Calamari with the LSTMs seems to take context successfully into account and outperforms Tesseract by far (CER of 1 vs. 6).

8 QUALITATIVE EVALUATION

Tables 4 and 5 show two small extracts from the ISRI Analytic Tools for the books *pilane* and *maungo*. In the former, it is clear that incorrect line segmentation plays a significant role in reducing the score. In the latter, none of the errors are caused by line segmentation errors, instead being the result of OCR misclassification mostly based on single characters. This, as well as the lesser amount of error counts, lead to a much reduced error rate.

Further analysis shows a clear difference in the nature of errors between Tesseract and Calamari. With Tesseract, single character confusions like quotation marks, digit '1' vs. lower-case letter 'l' vs. upper-case letter 'I' were some of the largest sources of errors—particularly in *nyalo*. The Calamari output suffers more frequently from incorrect line segmentation, resulting in entire lines of text missing from the output. The most common case is two consecutive lines being kept together during line segmentation. Another frequent error with the Calamari

https://doi.org/10.18489/sacj.v32i2.707

Table 3: CER scores for all four systems on post-processed data, including their average and the standard deviation of the scores for each system. Bolded scores indicate the best score of all systems for a given document across each row.

Book	Tess/IMU	Tess/nlbin	Calam/IMU	Calam/nlbin
maungo	0.42	0.42	**0.24**	0.50
baori	0.95	0.78	0.83	**0.63**
pilane	1.16	**0.73**	2.19	1.08
botshelo	1.08	0.96	**0.78**	1.14
sebele	2.33	2.15	**0.33**	0.93
ke ne ka	2.25	**1.44**	2.13	1.62
lesogana	2.32	**1.71**	2.64	2.53
nyalo	6.02	5.94	**1.07**	1.17
rammônê	5.16	**3.43**	4.97	6.46
AVG	2.41	1.95	**1.69**	1.78
STDEV	1.94	1.76	**1.50**	1.85

pipelines is missing spaces between words.

Although incorrect line segmentation results in relatively larger reductions in accuracy compared to single character errors, they might be easier to identify by a post-editor, as the error is often blatant. Character substitutions where characters are visually similar offer a challenge, also during human post-editing, and can more easily be missed by a human post-editor compared to blatant errors. Similarly, correcting large contiguous errors (e.g. with empty output for incorrect line segmentation) might also be easier, since it involves typing a few lines of running text which is faster per character than correcting the corresponding number of single character errors. For example, correcting a complete missing line of 50 characters requires the post-editor to identify one (large) error, to position the editing cursor once, and to type 50 contiguous characters. Alternatively, correcting 50 single character errors requires 50 errors to be identified, and the cursor to be positioned 50 times before correction. In our estimation, the output of Calamari therefore holds time savings in the post-editing phase for a project like ours.

The accurate handling of opening and closing quotes also offers advantages for post-processing, as it offers more information for accurate paragraph segmentation—an unmatched opening quotation mark strongly hints that the following lines should be joined with the currently processed paragraph.

9 DISCUSSION AND FUTURE WORK

Addressing the research questions, we can state the following:

Table 4: An extract (slightly adapted for clarity) from the log file created by the ISRI Analytic Tools for OCR Evaluation after evaluation on the book *pilane* using Calamari/IMU and after correcting for quotation marks, sorted by descending order of errors.

Error count	Correct	Generated
86	{di ne di sa ka ke tsa to...}	{}
84	{}	{-}
80	{gagwe. Go ka se ke ga re...}	{:::: < \n > }
76	{ }	{}
74	{o ise o omelele. Ka tsel...}	{}
70	{golwana tsa tsone di itl...}	{}
65	{go di tlhoa gotlhe-gotlh...}	{::::}

Table 5: An extract (slightly adapted for clarity) from the log file created by the ISRI Analytic Tools for OCR Evaluation after evaluation on the book *maungo* using Calamari/IMU and after correcting for quotation marks, sorted by descending order of errors.

Error count	Correct	Generated
39	{}	{-}
26	{ < \n > }	{ }
26	{}	{ }
11	{,}	{}
8	{ }	{ < \n > }
7	{ }	{}
6	{.}	{}

1. Everything else being equal, Calamari's average CER is around 20.41% better than Tesseract. This of course does not take into account the generalisability of the approach, where performance may not always correlate with certain features of the book such as its visual quality.

2. Combining different components in the pipeline has led to four different systems being compared to each other, one of them the combination reported in Kotzé and Wolff (2017). When applied to our data set, after post-processing normalisation, our best approach (Calamari/IMU) outperforms the original combination (Tesseract/IMU) by 30.01% CER on average.

3. A certain approach can be said to be more generalisable if there are not as many unexpected results on different inputs. For example, if apparently good quality scans lead to a notably worse result with a given system than similar scans have, the robustness of the approach and thus its generalisability over different data sets have to be called into

question. A smaller standard deviation of scores across a data set—even if the average may be worse—may be an indication that the system generalises better over a variety of different inputs. On average, Calamari/IMU has both the smallest standard deviation and the best CER, where the scores outperform the second best by around 14.52% and 5.48%, respectively.

However, unanswered questions remain. For example, with *pilane*, we found that the combination of Calamari/IMU is unexpectedly worse than all three other systems, even though we judged the quality of the scanned document as relatively good. Closer inspection reveals that the quality of the line segmentation is below par, even though visually, IMU has seemingly done a good job. A better understanding of the factors that impact the quality of line segmentation is therefore yet to be determined.

Digitisation projects involving books with more involved page layout might benefit from an additional step where page segmentation distinguishes between text and non-text regions of all pages. Software such as Aletheia (Clausner et al., 2011), Agora (Ramel et al., 2006), and LAREX (Reul et al., 2017) offer solutions to this problem. In our current document collection there are very few documents with any graphics, so this was not a pressing need for us.

We did not use Calamari to its full extent in the work presented in this paper. Particularly the use of model voting and retraining holds promise for further improvements in character recognition accuracy. Domain and genre adaptation through retraining or training from scratch would be beneficial to investigate. For the sake of high accuracy on mixed language documents it might also be beneficial to have multiple languages represented in the training data. It is unclear at this stage how much the domain and language of the training data influence the results, but such an investigation might provide some insights in the role of the LSTM layers in the neural network. They have shown their value in language modelling tasks, and combined with in-domain training data might provide further improvements. These possible improvements will, however, not reduce the extent to which line segmentation currently reduces the accuracy of the full Calamari pipeline.

For the Calamari pipeline, the line segmentation is the most pressing issue that needs improvement. It seems that Calamari is best at line level OCR on correctly identified lines. Since the recently released Tesseract 4 implements an LSTM approach similar to Calamari, it might prove an optimal solution when combined with its superior page analysis. Where the bad performance on a single book affects our evaluation of which system is best, clearly continuous evaluation on our growing collection will be important.

ACKNOWLEDGMENTS

The work described in this paper was supported by funding from the erstwhile Academy of African Languages and Science Strategic Project of the University of South Africa. We thank our former colleague Motswalle Kanyane for the quality assurance performed.

References

Breuel, T. (2008). The OCRopus open source OCR system. *Proceedings of IS&T/SPIE 20th Annual Symposium 2008*. https://doi.org/10.1117/12.783598

Clausner, C., Pletschacher, S. & Antonacopoulos, A. (2011). Aletheia – An advanced document layout and text ground-truthing system for production environments. *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR2011)*, 48–52. https://doi.org/10.1109/ICDAR.2011.19

Graves, A., Fernández, S., Gomez, F. & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the 23rd International Conference on Machine Learning*, 369–376. https://doi.org/10.1145/1143844.1143891

Gupta, M., Jacobson, N. & Garcia, E. (2007). OCR binarization and image pre-processing for searching historical documents. *Pattern Recognition, 40*(2), 389–397. https://doi.org/10.1016/j.patcog.2006.04.043

Hocking, J. & Puttkammer, M. (2016). Optical character recognition for South African languages. *Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), 2016*. https://doi.org/10.1109/RoboMech.2016.7813139

Kotzé, G. & Wolff, F. (2017). Developing and evaluating a pipeline for Setswana OCR. *Proceedings of the 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*. https://doi.org/10.1109/RoboMech.2017.8261154

Patvardhan, C., Verma, A. & Lakshmi, C. (2013). Document image denoising and binarization using Curvelet transform for OCR applications. *Proceedings of 2012 Nirma University International Conference on Engineering (NUiCONE)*. https://doi.org/10.1109/NUICONE.2012.6493228

Ramel, J., Busson, S. & Demonet, M. (2006). AGORA: The interactive document image analysis tool of the BVH project. *Proceedings of Second International Conference on Document Image Analysis for Libraries (DIAL'06)*. https://doi.org/10.1109/DIAL.2006.2

Reul, C., Springmann, U. & Puppe, F. (2017). LAREX – A semi-automatic open-source tool for layout analysis and region extraction on early printed books. *CoRR, abs/1701.07396*. https://doi.org/10.1145/3078081.3078097

Rice, S. (1996). *Measuring the accuracy of page-reading systems* (Ph.D. dissertation). University of Nevada, Las Vegas.

Smith, R. (2007). An overview of the Tesseract OCR engine. *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*, 629–633. https://doi.org/10.1109/ICDAR.2007.4376991

Wick, C., Reul, C. & Puppe, F. (n.d.). Calamari – A high-performance Tensorflow-based deep learning package for optical character recognition.