GÖTEBORGS UNIVERSITET
INST FÖR SVENSKA SPRÅKET

GU-ISS-2016-01

A free cloud service for OCR / En fri molntjänst för OCR

Project report

Lars Borin
Gerlof Bouma
Dana Dannélls

Forskningsrapporter från institutionen för svenska språket, Göteborgs universitet
Research Reports from the Department of Swedish

ISSN 1401-5919

www.svenska.gu.se/publikationer/GU-ISS

1 Introduction

The project *En fri molntjänst för OCR* ('A free cloud service for OCR'), funded by the National Library of Sweden (51-KB709-2012), ran from 1st September, 2013 until 31st August, 2014. The project was a collaboration between the University Library of University of Gothenburg (henceforth: UB) and Språkbanken at the Department of Swedish of University of Gothenburg (henceforth: SB), its aim to create a prototype Optical Character Recognition (OCR) web service for processing old Swedish texts that are printed in a blackletter (fraktur) or roman typeface, using one of two open source OCR engines. Our ultimate goal is to provide a service for libraries, museums and archives to upload any digitized document and retrieve an OCRed text with high quality, independent on the quality of the print. The project reported in this document is to be considered as the first steps toward this goal.

2 Background and motivation

Optical Character Recognition (OCR) is the process of transforming digital pictures to machine readable, editable and searchable text. Solutions have existed commercially since the 1950s, and since then OCR is widely used at the professional as well as the consumer level. Today there exist several services for smart phones where it is possible to easily take a photo of a document which can then be uploaded via an application to a server that interprets the picture and returns a text document. The quality of the services vary depending on how the picture was taken. The availability of such services is however high.

In many institutions like libraries, archives and museums there are ongoing initiatives to digitize in-house collections and archives. It is desirable to convert the text in all documents that have been produced, regardless of their form. One of the main motivations for this endeavor is to make the text searchable – both internally, within the institution, and externally, by web search engines. Another motivation is to promote scientific research and studies, as the availability of large amounts of material opens new possibilities for quantitative research, drives new learning, improves discovery techniques and facilitates development of language processing technology tools for historical digitized texts. Although individual pictures can easily be uploaded and OCR interpreted on for example a telephone, accurate OCRing of large amounts of digitized textual data is a demanding enterprise.

The quality of OCR depends greatly on the quality and type of material processed. For modern documents with good print quality, OCR results are typically very good. In such cases, almost error-free OCR is possible. For older texts the situation is different, however. Irregularities in the print (e.g., uneven inking and "wavy" baselines), the use of blackletter and older roman typefaces, deterioration of the paper, and variation in the language may all affect the accuracy of off-the-shelf OCR.

One of the best known commercial systems is Abbyy Finereader,[1] which is a high quality system in use in many institutions in Sweden, including UB. However, since it is a closed source commercial system, the ways in which we ourselves can use Abbyy to study how we might improve OCR quality for older material are severely limited, as would be dissemination of resulting experience and knowledge and our ability to use any results to offer an open OCR service to third parties.

There are however also high quality open source solutions. The most prominent of these is Tesseract,[2] which started out as a closed source system at HP Labs in the 1980s. After its release as open source, development has been supported by Google (Smith, 2007). OCRopus[3] is a relative newcomer, but with promising performance. Originally related to Tesseract, OCRopus now uses a completely independent neural-network based method to perform robust OCR. The evaluation and comparison of these two open source alternatives as the basis of our online service will be detailed in the following sections.

UB[4] was among the first libraries to digitize its catalog, already in the middle of the 1990s. The team at UB has extensive hands-on experience of large-scale digitization, including OCR. Today, OCR is part of the post-processing chain for all printed text digitized at UB.

SB[5] was established in 1975 as a national center for collection and processing of language resources,[6] primarily for Swedish text corpora and lexicon resources. It combines a high level of scientific expertise in language technology with a high level and specialised technical expertise and a well-developed technical infrastructure. Because of the specific needs of different hardware and software solutions, Språkbanken maintains its own servers for different purposes.

3 Activities and achievements

The reporting below of the project results is organized into five sections. The last of these sections contains pointers to the website and download locations for the output of the project.

3.1 Reference texts (SB/UB)

We have digitized a selection of 42 (mainly) blackletter texts printed between approx 1600 and 1800 from the collections of UB. The works differ in quality and clarity of the print, both in terms of the original print and how well the works have preserved, although no really deteriorated material was used. Most of the works contained a mixture of typefaces: a typical configuration is to set body text in fraktur, highlights in schwabacher, and names as well as (Romance) loan words in (italic) roman type. Apart from the mixed type faces, a problematic aspect for using off-the-shelf OCR, particular to Swedish prints from this time is the alphabet used. The prints contain ligatures that are rare or non-existent in modern text and are often not handled by OCR engines. In addition, Swedish blackletter print uses ‘aͤ’, ‘oͤ’ and ‘å’. The latter is typically not included in a fraktur character model, and is in addition very easily confused with the first, especially in deteriorated print.

[1]`<http://finereader.abbyy.com/>`

[2]`<https://code.google.com/p/tesseract-ocr/>`

[3]During the course of our project, OCRopus development was dormant. However as of october 2014 the project has a new location and appears to be more actively maintained. Old site: `<https://code.google.com/p/ocropus>`. New site: `<https://github.com/tmbdev/ocropy>`

[4]`<http://www.ub.gu.se/>`

[5]`<http://spraakbanken.gu.se/>`

[6]`<http://spraakbanken.gu.se/eng/resources>`

From the digitized texts, we randomly selected samples each consisting of several consecutive pages, for a total of 199 pages, which were transcribed externally by a company offering double-keying services (GREPECT, `<http://www.grepect.de>`). The guidelines we developed for transcription are relatively simple, but do contain instructions for ligatures, typeface changes and some rudimentary type size changes. On the basis of our inspections of the transcriptions, we consider them to be of very high quality. The full scanned material and the transcribed selection is available for download.

We divided the transcribed selection into a development set of 130 pages and an evaluation set of 69 pages. The former is used in the work described in section 3.2, the latter in the work described in section 3.4.

Further works used in development were Olof v. Dalin's *Swänska Argus* (1732–1734, Stockholm) and Peter Forsskål's *Tankar om Borgerliga Friheten.* The images of *Argus* were taken from the *Litteraturbanken* website (`<http://litteraturbanken.se>`), they were created by the National Library of Sweden. Our transcriptions are slightly edited versions of earlier transcriptions made by Carin Östman and Mats Thelander (Uppsala University). Our transcriptions include a smaller number of error corrections, and in addition we made separate versions for (part of) the original 1732–1734 edition and for the 1910/12 scientific edition (see below). The versions of course differ in pagination, but also show sporadic differences at the text string level. The scans of *Tankar* were made by UB as part of the present project. The accompanying transcriptions are based of transcriptions available at Projekt Runeberg (`<http://www.runeberg.org>`), which we corrected and adjusted for our project.

For comparison, we also considered some historical text in modern print. We used scans of a scientific edition of *Argus* (– Första Delen; Hesselman & Lamm, eds, 1910/1912, Svenska vitterhetssamfundet; scan taken from `<http://litteraturbanken.se>`) and *Stockholms stads tänkebocker från år 1529, del XV, 1626* (Wikstöm, ed, 1990, Stockholms stadsarkiv; scan from UB). For the development of a Swedish alphabet character model for the OCR engine, we sampled lines of scanned text and their transcriptions from works available at the Projekt Runeberg website – more on this dataset in the following section.

Part of the data mentioned here will be available from SB's servers, as detailed in section 3.5. We will supply links to data that we cannot, or are not allowed to, distribute.

3.2 Evaluation of OCRopus and Tesseract, development and evaluation of target material adapted language models (SB)

3.2.1 Initial, off-the-shelf evaluations

We started by evaluating the available models for fraktur of OCRopus v0.7 and Tesseract v3.02.

The version of OCRopus we used is an OCR system that does not rely on any explicit language information like word lists, character combination statistics (n-grams), etc. However, its fraktur model was trained on the German alphabet, which means it had no knowledge of 'å'. In addition, OCRopus comes with a model for modern print trained on an English alphabet (no 'å', 'ä', 'ö'), and a model that can read mixed fraktur and roman type text (no 'å'), but without the ability to say which typeface is being used.

A Tesseract OCR model is a combination of information about the alphabet/characters and the language being read (primarily in the form of word lists). There are models for modern Swedish and there is a fraktur model for Swedish, which appears to be a combination of a

German character model and a (modern) Swedish word list.

There are many parameters in the evaluation of OCR output that influence its outcome and several ways of counting errors. As part of the project we wrote software to evaluate OCR output according to two well-established measures. Given a page of OCR output and a (correct) transcript of that page, we report the *character error rate* (CER), which is the minimal number of character corrections (change, insert or delete one character) one has to make to completely correct the OCRed text according to the transcript, and *word error rate* (WER), analogously the minimal number of word corrections (change, insert or delete a complete word) one has to correct the OCRed output. These measures are then expressed as fractions of the size of the correct transcript, which means that it is possible to get an error rate of more than 100%.[7] The evaluation scripts are rather strict in that even mistakes in punctuation are counted. Material like headers, sidenotes and footnotes are ignored, as these are typically rearranged or even missing from the transcripts. Implicit in this kind of evaluation is an alignment of the OCR output and the transcript, where as many characters/words in the OCR output as possible are matched to their counterparts in the transcript, and vice versa. This alignment cannot only be used for counting errors, but also for inspection of the kind of errors produced, creation of training data for post-processing models (see section 3.2.3) and creation of training data for character models (see section 3.2.4).

The two measures CER and WER highlight different aspects of the performance. Performance at word level might generally be the more interesting of the two measures, as it is the word level that is relevant to indexing, searching and further computational linguistic processing. However, character level performance is often more indicative of the extent to which the result is humanly readable: one character error in each word will not impact readability too severely – reflected by moderate CER (inversely proportional to the average word length) – but the WER corresponding to this scenario will typically be very high.

Results of evaluating off-the-shelf OCRopus (mixed fraktur and roman type model) and Tesseract (Swedish fraktur model) are in table 1. As one can see, OCRopus outperforms Tesseract by a large margin, both in the *Argus* data (Volume I, issues XVI–XXV; 70 pages, 105k characters) and our mixed samples corpus (130 pages, 180k characters). This difference is far too large to be ascribed to the fact that we used a Tesseract model that does not handle roman type. We also see that both systems fare worse on the Mixed samples data, most likely due to the relatively stable quality of the Argus images. Still, we note that even for the OCRopus system, WERs lie around 50%, which roughly means that every other word contains a mistake.

The large difference between Tesseract and OCRopus suggests that the latter is a far better choice for an online OCR service for blackletter print. Before we dismiss Tesseract, however, we consider its performance after a bit of training and adjusting. The rows marked 'Modified Tesseract' in table 1 give results for Tesseract with a retrained character model and with a vocabulary compiled from Modern Swedish (*nysvenska*, 1526–1880) lexica and corpora available at SB. As one can see, the downloadable model of Tesseract is the best we seem to be able to get with this level of effort. Because of this, we have chosen to use OCRopus for

[7]Two extreme cases: On the one end, the OCR output and the transcript may be identical, in which case we do not have to make any corrections, giving an error rate of 0%. On the other extreme, the OCR output does not share a single character/word with the transcript, in which case number of errors corresponds to deleting the complete contents of the OCR output and adding (inserting) the complete contents of the transcript. Counting one error for each deleted and each inserted character/word, and dividing this by the length of transcript will give an error rate of 100% or above.

Data	Engine	CER (%)	WER (%)
Argus	Off-the-shelf OCRopus	10.4	47.2
	Off-the-shelf Tesseract	39.1	76.9
	Modified Tesseract	48.1	NA
	Modified OCRopus	11.7	49.1
Mixed samples	Off-the-shelf OCRopus	14.0	53.8
	Off-the-shelf Tesseract	37.6	72.3
	Modified Tesseract	40.2	NA
	Modified OCRopus	16.8	59.7

Table 1: Initial evaluation results for OCRopus and Tessearct.

the online service.

Rather than being one monolithic program, OCRopus is a suite of utilities to handle different aspects of OCR. There is software to handle image pre-processing (binarization), to do basic layout analysis (segmentation), to do OCR, to train new character models, and to compile HTML files according to the hOCR-standard. There are three downloadable character models available: for fraktur, for mixed fraktur and roman, and for modern print. The work that we conducted during the projekt and which is presented below, focusses on the core OCR parts, that is, the OCR engine itself, the training and the character models. We use the rest of the OCRopus tools to set up a working pipeline and webservice, but we have not explicitly evaluated their performance nor have we attempted to make improvements in those parts of the suite.

3.2.2 Retraining OCRopus character models

Since the downloadable fraktur models for OCRopus target the German alphabet, we have trained new character models that can recognize ‘å’, and which additionally distinguish between long-s ‘ſ’ and short-s. Although part of the software needed for training exists as part of OCRopus, we needed to create training data, consisting of images of lines of text and the transcriptions, and design a training regime for these new models. Based on a brief description (Breuel, A. Ul-Hasan, & Shafait, 2013) of the process, we developed software that creates training data using outline fonts – this involves composing a text with one/several appropriate fonts and applying some distortions that may occur in scanned texts like noise, stretching and line-warping.[8] We found that training OCRopus on a mix of synthetic and real data, the latter taken from *Tankar*, gives models that approach the quality of the downloadable ones, with the added benefit of being adapted to the Swedish alphabet. A total of 200 thousand lines of training data was used. In the future, we intend to use this software to improve upon our current fraktur model and to create models for other styles.

In addition, we wrote an extension to OCRopus that allows one to combine models for different typefaces. Although OCRopus comes with a model that can handle a mix of fraktur

[8]As mentioned, activity at the OCRopus development site was low during the project but has increased again after our project ended. The OCRopus authors have now released their own scripts implementing the techniques described in the cited works and used by us.

and roman type, that model is not able to indicate which typeface is being used and it is incompatible with the newly developed Swedish alphabet fraktur model. Using our extension, we can take two different character models (typically: one for fraktur and one for roman type), and combine their output, so that we can see which model is used where. Because of the mixed typefaces in the material, being able to switch typeface is beneficial for the output quality. The extension can trivially be updated to allow more typefaces (for the early print in particular Schwabacher).

In table 1, the results of the OCRopus extension and the new character model are given under the row 'Modified OCRopus'. We see that the modified system approaches the off-the-shelf system, but is not quite as accurate. In the future we hope to close this gap by further adjustment of our character training regime. Whether the current drop in accuracy is acceptable depends on how important the occurrence of 'å' in the OCR output is to the user.

3.2.3 Development of post-processing language modelling for OCRopus

Because OCRopus does not have any built-in way of exploiting language specific data to help OCR, we decided to apply a post-processing step to the OCRopus output. The method is based on a proposal by Kolak and Resnik (2005), and uses a combination of statistical knowledge of the type of character mistakes OCRopus makes (a so called *error model*) and knowledge of the language of the original document (a language model). Software to build error models by comparing OCRopus output and manually transcribed pages was developed in the project, using the OCR output and transcript alignments discussed in section 3.2.1. We also experimented with different types of language models, and will report here on the use of statistics over character sequences (*n-gram language models*), and statistics over full word occurrences.

We start by building an error model on 5 pages from *Argus* volume I not included in our development set. This may seem like a low number of pages, but as each page will contain over 2000 characters this gives us enough information to get character error estimates. Experiments with a larger selection of pages have not given better results. We also construct a trigram character model from *Argus* volume II (again, not included in the development evaluation set). The post-processor takes OCR output lines and construct hypotheses of what the original text might have been. The hypothesis that is chosen is the one that balances being a likely string in the language of the document and being likely to be (mis-)interpreted by the OCR engine as the output line.

The effect of applying this post-processor to the output of our Modified OCRopus is given in table 2, the lines marked with 'w/ trigram stat(istic)s Argus II'. As one can see the results are mixed. The CERs in both the *Argus* evaluation set and the Mixed samples evaluation set is higher than the Modified OCRopus models, however on the *Argus* test set, the WER is much lower after post-processing. The WER after post-processing in the Mixed samples evaluation set is slightly higher, however. We can explain these results in the following way: the postprocessing method has a tendency of deleting sequences of characters too easily when constructing a string hypothesis. This leads to increased CERs even if the model is capable of correcting character errors elsewhere. The improvement in WER in the *Argus* data shows that the model in principle is able to improve recognition when it doesn't show this sequence deleting behavior. Recall that CER and WER respond differently to the same kind of error: a concentration of character errors will affect CER as much as character errors that are evenly spread out. However, the former will not affect WER to the same extent as the latter. We were

Data	Engine	CER (%)	WER (%)
Argus	Modified OCRopus	11.7	49.1
	w/ trigram stats Argus II	15.0	40.2
	w/ trigram stats Argus vol II and Runeberg word stats	14.7	40.8
Mixed samples	Modified OCRopus	16.8	59.7
	w/ trigram stats Argus vol II	24.2	60.3
	w/ trigram stats oracle	17.7	55.2
	w/ trigram stats Argus vol II and Runeberg word stats	25.4	58.9

Table 2: Postprocessing evaluation results for OCRopus (fraktur)

not able to fix this unfortunate behavior of the post-processor in the course of the project, and intend to look at this as part the continuation of our development of the OCR service. The second discrepancy is that WER is not improved in the Mixed samples data. This shows the sensitivity of the language model to the application domain, a phenomenon well-known – but ill-mastered – in the language technology literature. Recall that the language model was created on *Argus* volume II. Although this data was not part of the Argus evaluation data, it is extremely similar to it, as it was written by the same writer, using the same conventions, on similar topics, close in time, etc. The Mixed samples data is spread out on all of these dimensions. To support this error analysis, we also supply evaluation using a 'biased' language model, namely a model constructed on the Mixed samples set itself. Note that this model will not be a perfect fit for any given document in the evaluation set, exactly because it is so mixed, but it will be a much better fit than the *Argus* volume II based model. Indeed, using this model we can see in the row marked 'w/ trigram stat(istic)s eval(uation data)', we see that the post-processor is able to improve WER. This should not be taken as an indication of the real performance of the post-processing method, but rather as a demonstration of what is wrong with the current language model. What we can say, however, is that the effectiveness of using the current post-processing model depends a lot on how much a user document resembles the language model data. We have not yet been able to find ways of defining a language model to overcome this domain sensitivity.

Looking at the output of the post-processor, we have also noted that the post-processor produces many nonsense strings. This is not surprising per se; a trigram character sequence model tends to assign high scores to sequences that look 'wordlike', but there is no guarantee that they are words or even possible words of the language. We therefore experimented with a language model that combines trigram statistics and a word list with frequencies. The favored hypothesis is then one that balances forming a likely character sequence, being comprised of frequent words and being likely to be (mis-)interpreted as the OCR output string. The word frequency list is compiled from 500 works taken from the Projekt Runeberg website and contains approximately 700 thousand word forms. The result of applying this post-processing method to the Mixed samples data is in the bottom row of table 2. Although the WER improves a little bit over only trigram statistics, the difference is negligible. This is in spite of

Data	Engine	CER (%)	WER (%)
Argus (1910–12)	ABBYY	0.5	2.8
	OCRopus Runeberg	1.2	5.7
Tankebok (1990)	ABBYY	0.6	6.0
	OCRopus Runeberg	0.9	6.7

Table 3: Evaluation of OCRopus with Runeberg-based character model

the fact that the coverage of the word list on the Mixed samples data is pretty good, with 81% of the tokens in the evaluation data occurring in the word list. The same can be observed for the *Argus* evaluation data, were the results with and without Runeberg word list are basically the same.

3.2.4 Development and evaluation of a roman type OCRopus model

A smaller part of the project was devoted to evaluating the accuracy of OCRopus on roman type, especially with modern editions of older texts in mind. The roman type character model that can be downloaded with OCRopus is however trained on English material and does therefore not know about the characters 'å', 'ä', 'ö'. We therefore trained a Swedish alphabet model for roman type. For the training data, we took images and transcripts from Projekt Runeberg, making sure to restrict ourselves to the group of fully proofread works. Although the Runeberg data is divided into pages, it is not divided into lines, which is needed to train OCRopus. We circumvented this problem as follows: The pages were binarized and segmented into lines with software from the OCRopus suite, after which we used the default roman type model to OCR the complete collection. The resulting text lines can then be aligned to the transcripts – reinserting hyphens – so that we have a set of images of text lines and their correct transcript. About 200 thousand lines from about 500 different works where then sampled to form the training data.

The resulting model is evaluated on Volume I, Issues XVI–XXV of *Argus* (1910–12, 75 pages, 105 thousand characters), and on a selection from *Tänkebok's renskrift* (1990, 20 pages, 50 thousand characters). The language in *Argus* is 18th century Swedish, the language in *Tänkebok* is 16th century Swedish. The evaluation set contains high quality scans of clear and clean pages made with modern printing techniques. Footnotes, margin notes and headers were ignored in the evaluation. We compare OCRopus performance with the results of running a commercial solution, ABBYY Recognition Server 3.0, on the same material. As table 3 shows, OCRopus shows very good results on this data. As expected, ABBYY, which can be considered the current state-of-the-art, does even better here. Interestingly, the WERs for both systems come really close on the *Tänkebok* data. This is because inserting a faulty word break is a particular common mistake ABBYY makes on this data, leading to a high WER, as every single incorrect word break leads to 2 word errors. Weaknesses of the OCRopus model include the letter *w*, capital letters, figures (i.e., numerals), and italics. These can generally be linked to under-representation of these categories in the training data. We have not given evaluation figures for the OCRopus' default roman type character model, but note that since the incidence of 'å', 'ä', 'ö', 'Å', 'Ä', 'Ö' can be expected to be around 3–5%, the performance of this model on Swedish text is guaranteed to be worse than what is reported here.

We conclude that training OCRopus to give good character models is possible with relatively little human intervention, provided enough transcribed text images are available. As in the Projekt Runeberg data, the base data does not have to be specifically intended for OCR training. Future work on our roman type model should involve careful selection of training data to amend the problem areas mentioned above.

3.3 Web service development (SB)

Our modified version of OCRopus forms the basis of an online, free OCR service at `<http://demo.spraakdata.gu.se/ocr>`. The service is available through a web page as well as through a programming interface (a REST web service API). We accept images in a variety of formats (PDF, PNG, JPG or GIF) and return plain text OCR interpretations of the images. Depending on their wishes, a user can choose for a fraktur-only or a combined fraktur and roman type OCR, and they can choose to use post-processing or not. If the combined OCR is chosen, the output also contains simple tags to indicate which typeface OCRopus thinks was used.

The online service runs on Språkbanken's own servers, it was implemented in Python using the Flask web application framework.[9]

3.4 Evaluation of webservice (UB)

An independent evaluation of the web service was performed by the UB partners of the project. The quality of the OCR as well as the easy of use of the website and the programming interface was considered.

3.4.1 OCR quality

The accuracy of the OCR system was evaluated on the basis of 69 pages taken from 39 different works in the collection scanned earlier in the project (97k characters, 16k words). Instead of our in-house evaluation tool used during development, an independent existing tool was chosen, ocrevalUAtion, developed in the context of IMPACT's SUCCEED project, (Carrasco, 2014). The API was used to automate interaction with our web service.

The results are given in table 4. The accuracy without post-processing is as we would expected from the development phase, with character accuracy in around 15% and word accuracy around 49%. Unfortunately, we again see that post-processing only deteriorates the results. There is a large variation in the OCR accuracy, as can be seen from the minimum and maximum values given in the table. At its worst, an OCRed text may as many mistakes as, or even more than, the number of words in the original text (WERs of 97.9% and 112.5%, respectively). On the other hand, some material comes out with WERs as low as 14%, which is good considering we are dealing with historical print. From inspecting document-by-document OCR results, it is also clear that the earliest prints (from the 1600s) are interpreted with lowest reliability.

Some recurrent mistakes in the material are the following:

- one character is interpreted as several, similar characters, for instance 'f' as 'fF' or 'm' as 'nm';
- several words are written together as one or (less frequently) one word is split up;

[9] `<http://flask.pocoo.org>`

	CER (%)				WER (%)			
Method	Mean	Median	Min	Max	Mean	Median	Min	Max
None	15.7	14.1	4.0	43.2	50.0	49.0	14.4	97.9
Post-processing	23.0	21.7	8.9	50.0	60.1	57.5	31.1	112.5

Table 4: Independent evaluation results through the web service

	CER (%)		WER (%)	
Resolution	Mean	Median	Mean	Median
300ppi	15.28	12.90	46.92	47.33
400ppi.	14.79	13.16	45.61	46.06

Table 5: Evaluation of varying image resolution.

- capitals may be misplaced in the text stream, lines and decorative figures are mistaken for text, text flow in the layout is not managed correctly
- ‘å’ and ‘ä’ are confused
- ‘/’ is interpreted as ‘-’

Part of these problems are related to the layout analysis performed by OCRopus, an aspect of OCR we have not considered at all. However, other problems, like properly recognizing word boundaries and character confusion mistakes are problems we had hoped to be able to address using language modelling.

Image quality will have a great impact on the OCR results. One aspect of an image’s quality is its resolution. We performed a brief test to assess the impact of varying resolution on the OCR accuracy. For a selection of 20 documents, we submitted both 300 and 400 ppi versions to the online service. The results in table 5 show that, although the results for 400ppi are slightly better, the impact of a using 300ppi is only small. Using a lower resolution may be attractive to users as it reduces storage and transport loads.

3.4.2 Brief comparison to ABBYY’s fraktur processing

For comparison, UB ran a small selection of 15 documents (23 thousand characters) through a test version of ABBYY Recognition Server 4.0, which can handle blackletter print. We will compare these results to our systems. However, the evaluation set is very small due to restrictions of the test version, so this comparison should only be taken as an indication. The results in table 6 show that default OCRopus with its downloadable mixed fraktur and roman model outperforms ABBYY, both in terms of CER and WER. Our modified version with the Swedish fraktur model beats ABBYY with respect to CER but not WER. Also note that the error rates reported here are higher than those reported for the Mixed samples development set and the Mixed samples test set.

3.4.3 Website and API use

The API is well-documented and works well, and the service’s usability is good overall. A wish list for future improvement, however, includes: support for TIFF input, support for

Data	Engine	CER (%)	WER (%)
Mixed sample, 15 pages	ABBYY	37.6	70.4
	OCRopus	22.6	70.4
	Modified OCRopus	28.0	78.4

Table 6: Evaluation of OCRopus with Runeberg-based character model

more output types (for instance PDF, and OCR-particular XML, HTML or JSON types), task progress indication, and an improved interface when managing several pages at once. In addition to its failure to improve quality, a problem particular to the post-processing functionality is its slow speed: it may take 20 minutes for certain pages to process, making the service impractical in a production work flow. Speed without post processing is however good enough.

3.5 Dissemination (SB/UB)

The project was presented at the following events:

- *Digitalisera vidare*, December 2013, Uppsala
- *Biblioteksdagarna*, Maj 2014, Umeå
- *Workshop on Computational Processing of Historical Texts*, September 2014, Helsinki
- *Digitalisera, men sen då?*, November 2014, Stockholm

The material and tools developed in the project can be downloaded from the project website: `<http://spraakbanken.gu.se/eng/ocr>`. This includes patches to OCRopus, trained character models for Tesseract and OCRopus and tools to train and evaluate the OCR engines. Apart from the *Swenska Argus* transcripts, which we did not create ourselves and were not obtained from a fully open source, all reference and scanned material is downloadable here, too. The pilot cloud service and web API are completely free access to anyone, and can be found at: `<http://demo.spraakdata.gu.se/ocr/>`.

4 Discussion and future work

In the project *En fri molntjänst för OCR*, we have evaluated two open-source OCR engines and further developed one of them: OCRopus. Using OCRopus we have set up a open webservice for OCR that can handle Swedish blackletter print as well as roman type print. The service is easy to use and available to anyone. There is both a webinterface for human interaction and an API, to facilitate automated access.

We have trained character models suitable for Swedish text, that give good results, especially in case of the roman type models. The new blackletter model comes close to but does not improve on the previously availabe downloadable OCRopus model, though the latter is not adapted to the Swedish alphabet. A small comparison to a commercial solution for blackletter OCR suggests that the OCRopus based OCR pipeline can achieve better results for Swedish blackletter print.

An important premise in the project was the expectation that natural language technology and the use of digital language resources could help to improve recognition quality. We investigated this by building a post-processing module for the OCRopus suite that uses knowledge about the target language to try to improve the OCR quality. The chosen method has previously been shown to be effective in other settings. However, we were only able to gain minor and partial improvements over the default OCRopus installation. We can clearly see a domain effect in the effectiveness of this post-processing model, where the method successfully reduces WER in data that is very similar to the data used to train the language models. We have not been able to overcome this lack of robustness. The use of temporally relevant word lists in addition to the statistical language model led to neither improvement nor deterioration. Given the framing of the project, this comes as a disappointment.

We conclude that this part of the project results once-again underlines the well-known natural language processing fact that domain sensitivity is a pervasive and hard-to-deal-with problem which has existed in the digital humanities for decades.

Even though post-processing is available through the webservice, at the moment it is probably best to run OCR without it or access the service using REST calls, especially since post-processing is a very time consuming task.

We intend to keep improving the service, first and foremost by continuing to train better character models. We have in the course of the project gained some insight into what helps in this respect, and hope to be able to use this experience to good effect. As far as incorporating further language resources, we will have to look at the post-processing implementation and perhaps consider alternative approaches for using such resources in the OCR pipeline.

The speed and efficiency of the webservice is another area to focus on – for the sake of the users but also to utilize our computing resources more effectively. The post-processing model was already mentioned as being very slow, and a different method and/or implementation technique may help this. Another factor in processing speed is OCRopus itself, in particular the image pre-processing and layout analysis steps. Improving these, something that was outside the scope of the current project, would help make the service more usable.

References

Breuel, T. M., A. Ul-Hasan, M. A. A., & Shafait, F. (2013). High performance OCR for printed English and fraktur using LSTM networks. In *International Conference on Document Analysis and Recognition.*

Carrasco, R. C. (2014). An open-source ocr evaluation tool. In *Proc. of the first international conference on digital access to textual cultural heritage* (pp. 179–184). NY, USA.

Kolak, O., & Resnik, P. (2005). OCR post-processing for low density languages. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing (HLT/EMNLP).* Vancouver, B.C., Canada.

Smith, R. (2007). An overview of the Tesseract OCR engine. In *Proc. of the 9^{th} int. conference on document analysis and recognition (ICDAR)* (pp. 629–633). IEEE Computer Society.

Forskningsrapporter från institutionen för svenska språket, Göteborgs universitet
Research Reports from the Department of Swedish

ISSN 1401-5919

www.svenska.gu.se/publikationer/GU-ISS