Challenges OCR today: INEL experiences

Niko Partanen

31 January 2017

Contents

Introduction

Optical Character Recognition (henceforth OCR) tools are irreplaceable in efforts to digitalize printed and hand written materials. During the last years there have been multiple very important and large digitalization projects, run both by academic institutions and libraries, often in cooperation. This opens entirely new situation for the research on these languages, and also allows the speakers to access this content on their language in a way which have never before been possible. Since many projects have produced millions of scanned pages, the principal question is how we can use this data. Most of the time the first step is to find some solutions to perform OCR on the text. Within INEL project we have done a wide ranging evaluation of different tools currently available, and despite our satisfactory results one must also acknowledge the existence of several alarming bottlenecks, insuitable technical combinations and dead ends within the existing workflows.

Moreover, it must be recognized that mere OCR of the text is often not enough in order to provide materials ready for linguistic analysis, but further restructuring is necessary. Often it adds value to automatically extract more information than just the text, as many documents contain systematically information about the author, location and time, among many other metadata variables. Information about the position and layout for characters, lines and words on pages is also among such data. The need for this is of course entirely different depending from the text type, since it is entirely possible to represent a book as a linear text paragraphs, which is not possible for newspaper texts which are divided into more complex layout elements. As a more technical note, also the recognition confidence for individual characters could be very useful for further post processing. This naturally grows importance with languages which have less mature OCR environment. As a last introductory comment, it must be emphasized that OCR is one of the few

computational tasks where manual post-correction is still necessary in the majority of cases. In this paper different problems are identified, and as far as possible, alternatives are offered and escape routes envisioned.

Although computational linguistics often rely on command line tools, there are some tasks which are particularly unsuitable to be done in terminal. As mentioned, one of these is post-processing of OCR result. This task in almost all cases demands visual comparison between the recognized text and the original image. There are plenty of different tools which offer possibility to manually correct OCR'd text, and virtually all of them have very similar user interfaces. The page image is on the left, and the text field on the right, and moving cursor around the text field has some kind of correspondence on top of the image. This makes one wonder why so many projects have ended up to the solution to develope new tools for post-correction. From this point of view it is also necessary to recognize character or word location on the page as essential part of the OCR result, since it is impossible to imagine a post-correction workflow which would function without this information, albeit there are cases where page position information can also be considered as unnecessary to bring into later annotation phases.

Compared to this, the initial training of an OCR model is very well suited to command line environment, especially for the transparency and reproducibility of the training process.

In this paper I compare the OCR model development in two OCR tools, OCRicola and Abbyy FineReader. In OCRicola I have used Skolt Saami as the test language, and in Abbyy FineReader Desktop I have used cyrillic Dolgan script. The Skolt Saami language model is distributed with this paper. The Skolt Saami language model has been developed in 2015 on University of Helsinki, and Dolgan work has been done in the University of Hamburg during 2016. The majority of the work presented in this paper was carried on within the INEL project in 2016, although the main author has been using different OCR tools during the last few years.

Current solutions

What do I want?

Following attributes would ideally be present in the output format:

- Recognized word
- Coordinates on page
- Recognition confidence
- Some style information
- Information about used OCR system
  - Version
  - System settings
  - Model used (with version, ideally commit hash from version control)

The format itself is naturally trivial, but some kind of an XML seems to be the current standard, although expressing same information in other formats such as JSON would be similarly easy. However, often offered formats such as plain text files are clearly not satisfactory.

The most problematic of the desired information is the styling, since this is expressed in very complex manner in printed publication. By *style* I refer to the settings such as italic or bold font, titles etc. I believe some features such as paragraph change can be later analysed from the line positions on the page, and thereby it doesn't belong directly to the OCR result itself.

At the moment the OCR market are dominated by ABBYY FineReader and Tesseract. The first is used with good success in many large projects, for example in Fenno-Ugrica collection of Finland's National Library, but it is entirely commercial software which makes it very problematic from perspectives of open science and software. Tesseract, and its derivations such as OCRicola, are entirely open source and highly trainable, but they offer just the very core functionality of OCR recognition and thereby have to be used in integration

with different software, and at least in INEL we haven't found any good combination of tools which could offer a highly reliable workflow.

The main problem with ABBYY based solutions is that the two versions of this program, Desktop and Engine, are very stricktly apart and it is not possible to combine their functionalities. The main functionality which Abbyy Desktop lacks is XML export. This means there is no export format which would directly store the page position information. However, in the Desktop version it is possible to train new language models. I will cover this more in detail below, but within this the problem is that the models trained Abbyy Desktop cannot be used in Abbyy Engine, which would have needed XML export. Also it is certain that many users would not want to rely on desktop environment, but would like to automatize their work in different ways, which in principle could be done with Abbyy Engine. However, with rare scripts and endangered languages the current models are highly insufficient and some additional training is necessary in order to achieve satisfactory output.

It is a good question why Abbyy doesn't maintain the page position information. For a long time I was suspecting this was because of capitalistic business interests: there are some reasons why it is more lucrative not to offer XML export for the casual user. However, after using Abbyy for a longer while I've started to assume there is a softer reson. Maybe Abbyy doesn't know how to store user edited position information? When the user edits the text in Abbyy, there are moments where the word entirely loses the highlighting, as if the software would not have information about the position any longer. There are similar problems with the Revizor editor developed by National Library of Finland in Fenno-Ugrica project. In Revizor the information is usually kept, but especially at the line edges it seems to lose the knowledge about which words were originally on the other line. This all signals that is must be extremely difficult to store word position information in user edited text. The assumption is very logical in the sense that the user can easily copy and paste text around, delete some words and retype them, and somehow the software would still need to keep track of positions. If the task is too difficult for the industry leader, we may in a deeper trouble than we thought, as it is unlikely smaller Open Source projects could reinvent this task. On the other hand Machine Learning based tools are advancing now so fast that many tasks we currently think are challenging may find surprising solutions in the coming years. I remain optimistic that in the longer run the Open Source community will emerge as the winner. One demand for this is certainly to stop building a new software in every project and cooperate more across different parties.

In INEL we are currently working through Abbyy Desktop plain text file and HTML exports, which both keep well the paragraph structure and, with some manual maintenance, also page numbering. This disregards the page position information, which is suboptimal in every sense, but for now there have not been better alternatives.

Training an Tesseract/OCRicola model

Tesseract model training is based to the idea that an example images are generated from an example text and a font.

Training an Abbyy model

Abbyy model training is a black box from which something quite good usually comes out.

Digitalization of the text or the book

I have often encountered the thought that the page position information would not be needed. However, there are several situations where this information is very critical. For example, different page elements such as page numbers are very easy to recognize from the layout when their position is known, but tend to get very messily mixed with rest of the content when the position information is lost. The same goes with paragraph information, since **a paragraph** is basically a structure in text which is closely related to

the organization of line indent, which, when known, makes paragraph detection rather reliable. Detecting it from unstructured lines is, on the other hand, rather hopeless.

It can be argued that there are special cases where this information can be disregarded. For example, when working with texts like Bible or Koran the original verse structure is usually enough to distinguish different elements, and in many instances the sameness of the text in each version is very highly enforced. Thereby OCR can be seen as a mean to get the text, but not so much as effort to digitalize the entire individual book or manuscript. More rare and more subjective to read the text is, more valuable it is to be able to reproduce the original text or to easily match the recognized words to positions on the page. In this sense OCR is ultimately just an attempt to claim that on these pixels, or on these millimeters on the original page, we have representations of these specific characters. Especially with old typefaces or handwritten manuscripts the matter of interpretation tends to become more significant. In the Skolt Saami and Dolgan texts I compare here this was not the issue.

Ideal scenario

We can compare…

Web based solutions

Revizor…

Recognized text as data packages

Where should the recognized texts be stored? I would say that a public repository of some sort is an ideal solution.

Need for REST API

Ideally one could easily open a wanted text in an online editor, save the changes and use an API to get the newest version and append that to the public repository.

Challenges for language technology

OCR'd texts offer many important tasks for natural language processing. First of all, there is the need to transform highly idiosynchronic writing systems into same ideally phoneme level representation. Moreover, OCR'd text is often not split neatly to individual words, but somewhat specific form of tokenization has to be employed. In the same vein one could also ask for detokenization, since the result of OCR is often not really the sentences, but not really the word tokens either. It is something in between, and the ready tools may not be adjusted to work exactly with that.

Mere written text often needs some kind of a morphological analysis in order to be more usable for linguistic analysis. Thereby applying different tools to this end is highly desiderable. It can also be mentioned that many texts are attractive targets for named entity recognition (NER) since they often cover repeatedly same prominent historical events in different languages.

One particularly relevant tiny task I can foresee is the automatic matching of recognized text with the original pages. This can be done by comparing automatically the proofread text masses with the XML output with word coordinate information. The two texts are naturally different, as one variant is not proofread and contains many errors which never have been present in the other version. However, I believe this is a minor obstacle, and detecting the most similar pages (especially knowing that the following page normally continues

the text) should be very doable in immediate future. I have not found any existing solutions to this, which again may reflect the immaturity in the use of OCR'd research data in the fields where high quality text output is a necessary or high preference.

Meeting with language documentation

In corpus linguistics and language technology the texts are usually treated the target of investigation in themselves. When applied to the endangered languages, there is often need to personalize them further and add into metadata information about the writers and translators. Instead of variables such as publishing time and place there is obvious need to know writer's birth time and birth place, among other things, but already those tell quite a bit about the possible native dialect of the writer. Often this kind of information is stored already in public repositories such as Wikipedia, so it is one additional question how to employ these connections the most effective way.

In many endangered language communities the pool of active and prominent speakers is small. It is highly likely that the same individuals have been writing to local newspapers and even published longer pieces of prose, and have also been speaking in public events which have been recorded. Or they have ended up to be recorded by different linguists of ethnographers, resulting in archive items which can be relatively easily associated with each other. Similarly in case the writers have deceased it can be expected that many communities recognize and remember them.

One also has to take into account that as the number of available resources grow, it is often not realistic that the research is conducted with one specific corpus, but there are possibilities

Last it could also be speculated that with many languages the existing language documentation resources are the largest available body of texts in that language. In these cases one could also imagine scenario where the language documentation data could be directly used to create an OCR model. This is somewhat counterintuitive, but text is text, and if the match between phoneme representation in written and spoken varieties is close enough, which it often is with languages with new orthographies, the changes for this to succeed are high.

Research possibilities of the parallel texts

Within Fenno-Ugrica project it has clearly been one of the corner ideas to collect resources which exist in several languages spoken in Russia. Thereby there are now publications such as Four Battles, which are translated to nine different Uralic languages. A cursory Google search reveals that translations must exist also in different Turkic languages. When the books are compared the texts appear immediately useful, since they share exactly the same structure, often almost on sentence level with only small deviations. Extracting the parallel sentences can be done with customary command line tools (Perl, sed) and, for example, hunalign, which automatically detects matching sentences and outputs a format that can be used in further processing. However, aligning the sentences is currently somewhat time consuming and more elaborate workflows are desperately needed. That said, matching the sentences is rewarding and fast enough that doing it manually for individual books is entirely feasible.

However, there are also resources which are parallel in less obvious ways. There are historical events which have been covered in every newspaper in the Soviet Union and beyond. The data does not form the parallel sentences as such, but still those texts are thematically linked in very intriguing ways. Applying tools such as named entity recognition into this kind of data could be very interesting for both language documentation and language technology, as well as to ordinary users and wider research in related fields such as anthropology.

Even more abstract parallels can be drawn between spoken and written resources, since there are many stories and narratives which have been recorded in different retellings and also published multiple times. A good example of this is the folktale Zarńia Bözha Kań, which also connects to the Russian folktale Maša i Medved'. It is published in two different variants which are both stored in Komi Nebögain collection in Syktyvkar, and also has been recorded, told by Vasiliy Lytkin in 1957. Even more curiously, I've found

relatively similar narratives in recently published corpus of Beserman Udmurt. It is still unknown how this data can exactly be used, but as far as I see there are lots of open roads which can be explored.

Closing words

The usability of different workflows related to OCR and reuse of the materials is closely connected to copyright. From my perspective the best and clearest solution is done in Fenno-Ugrica collection, where everything is in Public Domain. This means that the data can be used in any possible way we can or will be able to imagine. Naturally the existence of data in public domain is often related to the age of the data. This is why many large text corpora exist now for pre-1920's data, and as the years go by this year will gradually keep advancing. That said, it creates a situation where almost none of the presenters in this conference will live to see the currently published texts to enter Public Domain. There are exceptions such as the texts released into Public Domain by Ivan Belykh. Recently the main author of this article also compiled that text into an R package which makes it easily available. Conversations have also been ongoing to add it into interfaces such as Korp.

It is easy to talk about the differences in the data different projects deal with and produce, but there is a lot that is essentially same in all OCR workflows. These similarities mean that most of the practices and solutions which work in one place will also work elsewhere. There is some kind of a boundary in