Tesseract 3.0.2 - Training Material | WP3

Author: Jesse de Does, Adam Dudczak, Tomasz Parkola. INL Internal Review: Katrien Depuydt

INTRODUCTION

Tesseract is probably the most widely used open source OCR application. The information here is mainly based on http://code.google.com/p/tesseract-ocr/wiki/ReadMe, the Tesseract manual page http://tesseract-ocr.googlecode.com/svn-history/trunk/doc/tesseract.1.html and the FAQ http://code.google.com/p/tesseract-ocr/wiki/FAQ. The description applies to Tesseract 3.02.

Install software

There are two packages to install, the engine itself, and the training data for a language.

Linux

Tesseract is available directly from many Linux distributions. The package is generally called 'tesseract' or 'tesseract-ocr' - search your distribution's repositories to find it. E.g. on a recent ubuntu or debian system, simply

```
sudo apt-get install tesseract-ocr
```

will install the program.
Packages are also generally available for language training data (search the repositories,) but if not you will need to download the appropriate training data at http://code.google.com/p/tesseract-ocr/downloads/list, unpack it, and copy the .traineddata file into the 'tessdata' directory, probably `/usr/share/tesseract-ocr/tessdata` or `/usr/share/tessdata`, depending on your distribution.

If Tesseract isn't available for your distribution, or you want to use a newer version than is available, you can compile your own (cf. http://code.google.com/p/tesseract-ocr/wiki/Compiling). Note that older versions of Tesseract only supported processing TIFF files and their language training data format is incompatible with the one which is used in 3.0.x.

Mac OS X

The easiest way to install Tesseract is through homebrew (http://brew.sh) . Once homebrew is installed, you can install Tesseract by running the command: `brew install tesseract`.

If you want to use language training data not included with the homebrew package, download the appropriate training data, open it with Finder, and copy the .traineddata file into the `/usr/local/Cellar/tesseract/<version>/share/tessdata` directory.

Windows

An installer is available for Windows from our download page. This includes the English training data.
If you want to use another language, download the appropriate training data, unpack it using 7-zip (http://www.7-zip.org/) , and copy the .traineddata file into the 'tessdata' directory, probably

```
C:\Program Files\Tesseract OCR\tessdata.
```

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.

Other Platforms

Tesseract may work on more exotic platforms too. You can either try compiling it yourself, or take a look at the list of other projects using Tesseract.

System requirements

Tesseract has a small footprint and will run on most recent hardware, even on mobile devices.

Documentation

Most relevant documentation can be found at the project website, http://code.google.com/p/tesseract-ocr/.

INPUT

Input IMAGE formATS

According to the manual page, most image file formats (anything readable by the Leptonica image processing library) are supported. The Leptonica project page (http://code.google.com/p/leptonica/) lists at least jpg, png, tiff, bmp, pnm, gif, ps, pdf and webp.

SUPPORTED LANGUAGES

Currently supported languages for version 3.02 are: Afrikaans, Albanian, Arabic, Azerbaijani, Basque, Belarusian, Bengali, Bulgarian, Catalan, Cherokee, Chinese (Simplified), Chinese (Traditional), Croatian, Czech, Danish, Dutch, Esperanto, Estonian, Finnish, Frankish, French, Galician, German, Greek, Hebrew, Hindi, Hungarian, Icelandic, Indonesian, Italian, Italian (Old), Japanese, Kannada, Korean, Latvian, Lithuanian, Macedonian, Malay, Malayalam, Maltese, Middle English (1100-1500), Middle French (ca. 1400-1600), Norwegian, Polish, Portuguese, Romanian, Serbian (Latin), Slovakian, Slovenian, Spanish, Spanish (Old), Swahili, Swedish, Tagalog, Tamil, Telugu.
Language data can be downloaded at http://code.google.com/p/tesseract-ocr/downloads/list. The uncompressed trained data should be copied to the TESSDATA directory.
Tesseract 3.0.2 supports recognitions of images containing text in more than one language. Users can specify[1] several languages and Tesseract will use the most accurate recognition as a result. Users need to keep in mind that recognition of pages in several languages last much longer than in case of one language profile.

LIMITATIONS

The fact that your image format is supported and your language is implemented does not necessarily mean that your recognition results will be satisfactory. The main reasons for suboptimal results are

- Poor quality images, for instance low-resolution black and white images from old micro-films
- Degraded documents (warped, unclear printing, damaged, …)
- Font shapes unknown to the engine

1 It can be using the following syntax: `tesseract inputFile outputFile -lang eng+pol`

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.

- Your language may be listed as supported, but the actual language in your documents may be incompatible with the implemented language support, if it contains specific terminology, historical or regional language.

EXTENDING LANGUAGE SUPPORT

One of the peculiarities of Tesseract is that glyph shape training data and language support data are tied up. This means that compiled word lists are part of the trained data bundle. A limited amount of words can be added without building a new data package, as a user word list. Otherwise, one has to retrain the engine (cf. relevant section). A workaround for the entanglement of language and font data is as follows[2]. Put the trained data file for your language in a separate directory. Now changedir to that directory. Assume the trained data file you start from is LANG.traineddata.

1) Unpack trained data
   ```
   combine_tessdata -u traineddata_file LANG.
   ```
2) Compile a word list to dawg format
   ```
   wordlist2dawg your_word_list new_dawg_file LANG.unicharset
   ```
3) Replace the word_dawg
   ```
   cp new_dawg_file LANG.word-dawg
   ```
4) Repack the trained data
   ```
   combine_tessdata LANG.
   ```
5) Install your file LANG.traineddata by copying it to the tesseract data directory[3].

OUTPUT

Output formats

There are two possible full text output formats: plain text and hOCR. **hOCR**[4] is an open standard which defines a data format for representation of OCR output. The standard aims to embed layout, recognition confidence, style and other information into the recognized text itself. Embedding this data into text in the standard HTML format is used to achieve that goal. Both are not entirely suitable for deployment in digital libraries, where one typically prefers XML-based solutions. Conversion of hOCR to ALTO or direct ALTO output is an obvious desideratum. No such utility seems to be available.
Another output format, which is relevant in the training process, is the box format, which gives bounding boxes for each recognized character.

2 Tesseract has two types of language support: dictionaries for single words, and support for bigram models. In the latter case (as in the default english language support) the procedure will probably not work.
3 Cf the installation instructions above to find out where that might be, depending on your OS/distribution.
4 Specification at https://docs.google.com/document/d/1QQnIQtvdAC_8n92-LhwPcjtAUFwBlzE8EWnKAxlgVf0/preview

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.

Procedures

RunNING OCR

The manual page is at http://tesseract-ocr.googlecode.com/svn/trunk/doc/tesseract.1.html.

SYNOPSIS

tesseract imagename outbase [-l lang] [-psm N] [configfile …]

OPTIONS

IMAGENAME

The name of the input image. Most image file formats (anything readable by Leptonica) are supported.

OUTBASE

The basename of the output file (to which the appropriate extension will be appended). By default the output will be named outbase.txt.

-L LANG

The language to use. If none is specified, English is assumed. Multiple languages may be specified, separated by plus characters. Tesseract uses 3-character ISO 639-2 language codes. (See LANGUAGES)

-PSM N

Set Tesseract to only run a subset of layout analysis and assume a certain form of image. The options for N are:

```
0 = Orientation and script detection (OSD) only.
1 = Automatic page segmentation with OSD.
2 = Automatic page segmentation, but no OSD, or OCR.
3 = Fully automatic page segmentation, but no OSD. (Default)
4 = Assume a single column of text of variable sizes.
5 = Assume a single uniform block of vertically aligned text.
6 = Assume a single uniform block of text.
7 = Treat the image as a single text line.
8 = Treat the image as a single word.
9 = Treat the image as a single word in a circle.
10 = Treat the image as a single character.
```

-V

Returns the current version of the Tesseract(1) executable.

CONFIGFILE

The name of a config to use. A config is a plaintext file which contains a list of variables and their values, one per line, with a space separating variable from value. Interesting config files include: hocr - Output in hOCR format instead of as a text file. If this configuration file is not present, you

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.

can create and use a plain text file containing the line

```
tessedit_create_hocr<TAB>T
```

Nota Bene: The options -l lang and -psm N must occur before any configfile.

TRAINING TESSERACT

Tesseract is retrainable. Documentation on the training process is available at http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3. A shell script implementing the training process is available in the appendix.
Though this takes care of the purely technical part of the process, it defines a way of compiling training data to Tesseract format rather than an approach to developing it with optimal recognition results.
The basic requirements for training a font/language combination are:

1) A combination of a (usually black and white) page image and a text file (box format) listing containing a number of lines with a character and the bounding box of an occurrence of that character in the page image. A box file contains lines like:

```
D 124 2906 150 2954 0
e 153 2908 171 2950 0
P 187 2900 237 2959 0
a 240 2908 263 2947 0
e 263 2909 285 2947 0
l 285 2909 299 2958 0
b 299 2911 327 2958 0
r 327 2901 345 2951 0
u 348 2910 378 2950 0
```

2) Word lists: a list of frequent words and a broader list aiming at a more comprehensive coverage of the language

3) Some small configuration files.

From this, command line utilities supplied with the basic Tesseract distribution can build a new trained data bundle. An example script for the process is given in the appendix.
This does not yet give us guidelines on how to proceed when we want to train for a specific collection. Ideally, one might hope that a set of images together with a set of ground truth transcriptions might be enough to train the engine. In practice, this is not so easy.
First, since there is no practical tool available to align the images with a plain text ground truth transcription, we need to enhance our ground truth with character bounding boxes or even bounding polygons.Moreover, there are some restrictions: character bounding boxes should not overlap; there should only be one font per training image/box file pair.
More importantly, while one might expect that damaged instances of a glyph shape might also be informative to the character classification process, this appears not to be the case. Tesseract assumes its training material to represent prototypical shapes rather than possibly noisy instances.
Several solutions have been developed to bridge the gap between ground truth data and a Tesseract trained data bundle.
User interfaces have been developed to create (or manually correct automatically created) box files, for instance JtessBoxEditor (http://vietocr.sourceforge.net/training.html) or web-based Cutouts (http://wlt.synat.pcss.pl/cutouts).

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.

We also mention two approaches based on the PAGE XML ground truth format (http://www.primaresearch.org/tools.php). This format allows user-friendly development of ground truth material with the option of specifying coordinates for text regions, lines, words and individual glyphs with the *Aletheia* tool (ibidem), which can be used freely for non-commercial purposes.

- The Poznan Supercomputing and Networking Center (PSNC) has developed two handy tools to automatically develop training data starting from an image and a PAGE XML ground truth file with glyph coordinate information. The first tool cuts out the glyphs from the image, creating individual images. After this stage, noisy character images may be re-moved. The second tool recombines the glyphs into a “cleaner” input image which can be used in the Tesseract training process, and also generates the required box file. The use of these tools is documented in the file `IC-Tesseracttrainingworkflow-200913-0919-9296.pdf`, included in the training package.

- In the EMOP[5] project, a tool Franken+ has been produced. Provided the binarised image and the resulting XML file generated with Aletheia, Franken+ extracts individual TIFF im-ages for each letter blocked-out using Aletheia, giving the user the opportunity to hand-pick the best instances of each letter (thus producing a "font" consisting of only hand-picked images). Using this font, Franken+ can then create synthetic TIFF images of text "printed" using this font, with corresponding BOX files, which are then used to train Tesseract OCR engine in order to OCR images of documents printed with the relevant historic font. Using these synthetic images and their corresponding BOX files, Franken+ then automates the Tesseract font training process and allows a user to test this font.

For a comparison between the FineReader and the Tesseract OCR trainability, cf. for instance the case study http://lib.psnc.pl/dlibra/doccontent?id=358, which we include with the current SUCCEED training materials for Tesseract.

FuRTHER READING

- Information on training Tesseract in the EMOP project at: http://emop.tamu.edu/node/47, http://emop.tamu.edu/node/54 for the Franken+ tool. Information on the Franken+ tool at: http://dh-emopweb.tamu.edu/Franken+/. Publication: Early Modern OCR Project (eMOP) at Texas A&M University: Using Aletheia to Train Tesseract , http://dl.acm.org/citation.cfm?id=2494304.

- The following IMPACT reports (included in training materials) describing, among others, experiences on using the Tesseract engine on a diverse set of historical documents.

    - The already mentioned *Report on the comparison of Tesseract and ABBYY FineReader OCR engines* from PSNC. PSNC instructions for the Tesseract train-ing process are included in the current training package.

5 http://emop.tamu.edu/

- *Lexicon-supported OCR of eighteenth century Dutch books: a case study* (also includes a comparison with Finereader on the same data)

Appendix

Tesseract training script

Building a TESSERACT TRAINED DATA BUNDLE

This bash script assumes the presence, in the current directory, of

1. A file 'files.lst' containing the base names of the images and box files (such that the list contains, for instance the line "image1" when image1.png and image1 are the respective image and box files)
2. For each line li in this file containing a string si, an image file si.$EXTENSION and a box file si.box.
3. A word list named words.list

```
# settings for variables
FONTNAME=antiqua

# paths with Tesseract's binaries and shared data
TESS_BIN=/home/jesse/opt/bin
export TESSDATA_PREFIX=/home/jesse/opt/share/
TESS_SHARE=/home/jesse/opt/share/tessdata/

# document specific settings
LANGNAME=emp # early modern polish

NAME=$LANGNAME.$FONTNAME.exp
EXTENSION=.png

echo "combined 0 0 0 0 1" >font_properties &&
$TESS_BIN/unicharset_extractor *.box &&
for x in `cat files.txt`
do
  echo "tesseract training $x$EXTENSION"
  $TESS_BIN/tesseract $x$EXTENSION$NAME$x nobatch box.train.stderr
done
rm combined.tr
cat *.tr >combined.tr
$TESS_BIN/mftraining -F font_properties -U unicharset -O $LANGNAME.unicharset combined.tr
&& $TESS_BIN/cntraining combined.tr || exit
mv pffmtable $LANGNAME.pffmtable
mv normproto $LANGNAME.normproto
mv inttemp $LANGNAME.inttemp
wordlist2dawg words.list $LANGNAME.word-dawg unicharset
```

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.

```
$TESS_BIN/combine_tessdata $LANGNAME.
```

Licensing

Tesseract is licensed under the Apache License, Version 2.0 (http://www.apache.org/licenses/LICENSE-2.0).

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.