NAME

tesseract − command−line OCR engine

SYNOPSIS

tesseract *FILE OUTPUTBASE* [*OPTIONS*]... [*CONFIGFILE*]...

DESCRIPTION

tesseract(1) is a commercial quality OCR engine originally developed at HP between 1985 and 1995. In 1995, this engine was among the top 3 evaluated by UNLV. It was open−sourced by HP and UNLV in 2005, and has been developed at Google since then.

IN/OUT ARGUMENTS

FILE

The name of the input file. This can either be an image file or a text file.

Most image file formats (anything readable by Leptonica) are supported.

A text file lists the names of all input images (one image name per line). The results will be combined in a single file for each output file format (txt, pdf, hocr, xml).

If *FILE* is stdin or − then the standard input is used.

OUTPUTBASE

The basename of the output file (to which the appropriate extension will be appended). By default the output will be a text file with .txt added to the basename unless there are one or more parameters set which explicitly specify the desired output.

If *OUTPUTBASE* is stdout or − then the standard output is used.

OPTIONS

−c *CONFIGVAR=VALUE*

Set value for parameter *CONFIGVAR* to VALUE. Multiple **−c** arguments are allowed.

−−dpi *N*

Specify the resolution *N* in DPI for the input image(s). A typical value for *N* is 300. Without this option, the resolution is read from the metadata included in the image. If an image does not include that information, Tesseract tries to guess it.

−l *LANG*, **−l** *SCRIPT*

The language or script to use. If none is specified, eng (English) is assumed. Multiple languages may be specified, separated by plus characters. Tesseract uses 3−character ISO 639−2 language codes (see **LANGUAGES AND SCRIPTS**).

−−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. (not implemented)
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.
11 = Sparse text. Find as much text as possible in no particular order.

12 = Sparse text with OSD.
13 = Raw line. Treat the image as a single text line,
bypassing hacks that are Tesseract−specific.

−−oem *N*
Specify OCR Engine mode. The options for *N* are:

0 = Original Tesseract only.
1 = Neural nets LSTM only.
2 = Tesseract + LSTM.
3 = Default, based on what is available.

−−tessdata−dir *PATH*
Specify the location of tessdata path.

−−user−patterns *FILE*
Specify the location of user patterns file.

−−user−words *FILE*
Specify the location of user words file.

CONFIGFILE
The name of a config to use. The name can be a file in tessdata/configs or tessdata/tessconfigs, or an absolute or relative file path. A config is a plain text file which contains a list of parameters and their values, one per line, with a space separating parameter from value.

Interesting config files include:

- **alto** — Output in ALTO format (*OUTPUTBASE*.xml).
- **hocr** — Output in hOCR format (*OUTPUTBASE*.hocr).
- **pdf** — Output PDF (*OUTPUTBASE*.pdf).
- **tsv** — Output TSV (*OUTPUTBASE*.tsv).
- **txt** — Output plain text (*OUTPUTBASE*.txt).
- **get.images** — Write processed input images to file (tessinput.tif).
- **logfile** — Redirect debug messages to file (tesseract.log).
- **lstm.train** — Output files used by LSTM training (*OUTPUTBASE*.lstmf).
- **makebox** — Write box file (*OUTPUTBASE*.box).
- **quiet** — Redirect debug messages to */dev/null*.

It is possible to select several config files, for example tesseract image.png demo alto hocr pdf txt will create four output files demo.alto, demo.hocr, demo.pdf and demo.txt with the OCR results.

Nota bene: The options **−l** *LANG*, **−l** *SCRIPT* and **−−psm** *N* must occur before any *CONFIGFILE*.

SINGLE OPTIONS

−h, −−help
Show help message.

−−help−extra
Show extra help for advanced users.

−−help−psm
Show page segmentation modes.

−−help−oem
Show OCR Engine modes.

−v, −−version
Returns the current version of the tesseract(1) executable.

−−list−langs
List available languages for tesseract engine. Can be used with **−−tessdata−dir** *PATH*.

−−print−parameters
Print tesseract parameters.

LANGUAGES AND SCRIPTS

To recognize some text with Tesseract, it is normally necessary to specify the language(s) or script(s) of the text (unless it is English text which is supported by default) using **−l** *LANG* or **−l** *SCRIPT*.

Selecting a language automatically also selects the language specific character set and dictionary (word list).

Selecting a script typically selects all characters of that script which can be from different languages. The dictionary which is included also contains a mix from different languages. In most cases, a script also supports English. So it is possible to recognize a language that has not been specifically trained for by using traineddata for the script it is written in.

More than one language or script may be specified by using +. Example: tesseract myimage.png myimage −l eng+deu+fra.

https://github.com/tesseract−ocr/tessdata_fast provides fast language and script models which are also part of Linux distributions.

For Tesseract 4, tessdata_fast includes traineddata files for the following languages:

afr (Afrikaans), **amh** (Amharic), **ara** (Arabic), **asm** (Assamese), **aze** (Azerbaijani), **aze_cyrl** (Azerbaijani − Cyrilic), **bel** (Belarusian), **ben** (Bengali), **bod** (Tibetan), **bos** (Bosnian), **bre** (Breton), **bul** (Bulgarian), **cat** (Catalan; Valencian), **ceb** (Cebuano), **ces** (Czech), **chi_sim** (Chinese simplified), **chi_tra** (Chinese traditional), **chr** (Cherokee), **cym** (Welsh), **dan** (Danish), **deu** (German), **dzo** (Dzongkha), **ell** (Greek, Modern, 1453−), **eng** (English), **enm** (English, Middle, 1100−1500), **epo** (Esperanto), **equ** (Math / equation detection module), **est** (Estonian), **eus** (Basque), **fas** (Persian), **fin** (Finnish), **fra** (French), **frk** (Frankish), **frm** (French, Middle, ca.1400−1600), **gle** (Irish), **glg** (Galician), **grc** (Greek, Ancient, to 1453), **guj** (Gujarati), **hat** (Haitian; Haitian Creole), **heb** (Hebrew), **hin** (Hindi), **hrv** (Croatian), **hun** (Hungarian), **iku** (Inuktitut), **ind** (Indonesian), **isl** (Icelandic), **ita** (Italian), **ita_old** (Italian − Old), **jav** (Javanese), **jpn** (Japanese), **kan** (Kannada), **kat** (Georgian), **kat_old** (Georgian − Old), **kaz** (Kazakh), **khm** (Central Khmer), **kir** (Kirghiz; Kyrgyz), **kmr** (Kurdish Kurmanji), **kor** (Korean), **kor_vert** (Korean vertical), **kur** (Kurdish), **lao** (Lao), **lat** (Latin), **lav** (Latvian), **lit** (Lithuanian), **ltz** (Luxembourgish), **mal** (Malayalam), **mar** (Marathi), **mkd** (Macedonian), **mlt** (Maltese), **mon** (Mongolian), **mri** (Maori), **msa** (Malay), **mya** (Burmese), **nep** (Nepali), **nld** (Dutch; Flemish), **nor** (Norwegian), **oci** (Occitan post 1500), **ori** (Oriya), **osd** (Orientation and script detection module), **pan** (Panjabi; Punjabi), **pol** (Polish), **por** (Portuguese), **pus** (Pushto; Pashto), **que** (Quechua), **ron** (Romanian; Moldavian; Moldovan), **rus** (Russian), **san** (Sanskrit), **sin** (Sinhala; Sinhalese), **slk** (Slovak), **slv** (Slovenian), **snd** (Sindhi), **spa** (Spanish; Castilian), **spa_old** (Spanish; Castilian − Old), **sqi** (Albanian), **srp** (Serbian), **srp_latn** (Serbian − Latin), **sun** (Sundanese), **swa** (Swahili), **swe** (Swedish), **syr** (Syriac), **tam** (Tamil), **tat** (Tatar), **tel** (Telugu), **tgk** (Tajik), **tgl** (Tagalog), **tha** (Thai), **tir** (Tigrinya), **ton** (Tonga), **tur** (Turkish), **uig** (Uighur; Uyghur), **ukr** (Ukrainian), **urd** (Urdu), **uzb** (Uzbek), **uzb_cyrl** (Uzbek − Cyrilic), **vie** (Vietnamese), **yid** (Yiddish), **yor** (Yoruba)

To use a non−standard language pack named foo.traineddata, set the TESSDATA_PREFIX environment variable so the file can be found at TESSDATA_PREFIX/tessdata/foo.traineddata and give Tesseract the argument **−l** foo.

For Tesseract 4, tessdata_fast includes traineddata files for the following scripts:

Arabic, **Armenian**, **Bengali**, **Canadian_Aboriginal**, **Cherokee**, **Cyrillic**, **Devanagari**, **Ethiopic**, **Fraktur**, **Georgian**, **Greek**, **Gujarati**, **Gurmukhi**, **HanS** (Han simplified), **HanS_vert** (Han simplified, vertical), **HanT** (Han traditional), **HanT_vert** (Han traditional, vertical), **Hangul**, **Hangul_vert** (Hangul vertical), **Hebrew**, **Japanese**, **Japanese_vert** (Japanese vertical), **Kannada**, **Khmer**, **Lao**, **Latin**, **Malayalam**, **Myanmar**, **Oriya** (Odia), **Sinhala**, **Syriac**, **Tamil**, **Telugu**, **Thaana**, **Thai**, **Tibetan**, **Vietnamese**.

The same languages and scripts are available from **https://github.com/tesseract−ocr/tessdata_best**. tessdata_best provides slow language and script models. These models are needed for training. They also can give better OCR results, but the recognition takes much more time.

Both tessdata_fast and tessdata_best only support the LSTM OCR engine.

There is a third repository, **https://github.com/tesseract−ocr/tessdata**, with models which support both the Tesseract 3 legacy OCR engine and the Tesseract 4 LSTM OCR engine.

CONFIG FILES AND AUGMENTING WITH USER DATA

Tesseract config files consist of lines with parameter−value pairs (space separated). The parameters are documented as flags in the source code like the following one in tesseractclass.h:

STRING_VAR_H(tessedit_char_blacklist, "", "Blacklist of chars not to recognize");

These parameters may enable or disable various features of the engine, and may cause it to load (or not load) various data. For instance, let's suppose you want to OCR in English, but suppress the normal dictionary and load an alternative word list and an alternative list of patterns — these two files are the most commonly used extra data files.

If your language pack is in */path/to/eng.traineddata* and the hocr config is in */path/to/configs/hocr* then create three new files:

/path/to/eng.user−words:

```
the
quick
brown
fox
jumped
```

/path/to/eng.user−patterns:

```
1−\d\d\d−GOOG−411
www.\n\\\*.com
```

/path/to/configs/bazaar:

```
load_system_dawg     F
load_freq_dawg       F
user_words_suffix    user−words
user_patterns_suffix user−patterns
```

Now, if you pass the word *bazaar* as a *CONFIGFILE* to Tesseract, Tesseract will not bother loading the system dictionary nor the dictionary of frequent words and will load and use the *eng.user−words* and *eng.user−patterns* files you provided. The former is a simple word list, one per line. The format of the latter is documented in *dict/trie.h* on *read_pattern_list()*.

ENVIRONMENT VARIABLES

TESSDATA_PREFIX

If the TESSDATA_PREFIX is set to a path, then that path is used to find the tessdata directory with language and script recognition models and config files. Using **−−tessdata−dir** *PATH* is the recommended alternative.

OMP_THREAD_LIMIT

If the tesseract executable was built with multithreading support, it will normally use four CPU cores for the OCR process. While this can be faster for a single image, it gives bad performance if the host computer provides less than four CPU cores or if OCR is made for many images. Only a single CPU core is used with OMP_THREAD_LIMIT=1.

HISTORY

The engine was developed at Hewlett Packard Laboratories Bristol and at Hewlett Packard Co, Greeley Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some C++izing in 1998. A lot of the code was written in C, and then some more was written in C++. The C++ code makes heavy use of a list system using macros. This predates STL, was portable before STL, and is more efficient than STL lists, but has the big negative that if you do get a segmentation violation, it is hard to debug.

Version 2.00 brought Unicode (UTF−8) support, six languages, and the ability to train Tesseract.

Tesseract was included in UNLV's Fourth Annual Test of OCR Accuracy. See **https://github.com/tesseract−ocr/docs/blob/master/AT−1995.pdf**. Since Tesseract 2.00, scripts are now included to allow anyone to reproduce some of these tests. See **https://github.com/tesseract−ocr/tesseract/wiki/TestingTesseract** for more details.

Tesseract 3.00 added a number of new languages, including Chinese, Japanese, and Korean. It also introduced a new, single−file based system of managing language data.

Tesseract 3.02 added BiDirectional text support, the ability to recognize multiple languages in a single image, and improved layout analysis.

Tesseract 4 adds a new neural net (LSTM) based OCR engine which is focused on line recognition, but also still supports the legacy Tesseract OCR engine of Tesseract 3 which works by recognizing character patterns. Compatibility with Tesseract 3 is enabled by −−oem 0. This also needs traineddata files which support the legacy engine, for example those from the tessdata repository (**https://github.com/tesseract−ocr/tessdata**).

For further details, see the release notes in the Tesseract wiki (**https://github.com/tesseract−ocr/tesseract/wiki/ReleaseNotes**).

RESOURCES

Main web site: **https://github.com/tesseract−ocr** User forum: **http://groups.google.com/group/tesseract−ocr** Wiki: **https://github.com/tesseract−ocr/tesseract/wiki** Information on training: **https://github.com/tesseract−ocr/tesseract/wiki/TrainingTesseract**

SEE ALSO

ambiguous_words(1), cntraining(1), combine_tessdata(1), dawg2wordlist(1), shape_training(1), mftraining(1), unicharambigs(5), unicharset(5), unicharset_extractor(1), wordlist2dawg(1)

AUTHOR

Tesseract development was led at Hewlett−Packard and Google by Ray Smith. The development team has included:

Ahmad Abdulkader, Chris Newton, Dan Johnson, Dar−Shyang Lee, David Eger, Eric Wiseblatt, Faisal Shafait, Hiroshi Takenaka, Joe Liu, Joern Wanke, Mark Seaman, Mickey Namiki, Nicholas Beato, Oded

Fuhrmann, Phil Cheatle, Pingping Xiu, Pong Eksombatchai (Chantat), Ranjith Unnikrishnan, Raquel Romano, Ray Smith, Rika Antonova, Robert Moss, Samuel Charron, Sheelagh Lloyd, Shobhit Saxena, and Thomas Kielbus.

For a list of contributors see **https://github.com/tesseract−ocr/tesseract/blob/master/AUTHORS**.

COPYING