NAME

text2image − generate OCR training pages.

SYNOPSIS

text2image −−text *FILE* −−outputbase *PATH* −−fonts_dir *PATH* [OPTION]

DESCRIPTION

text2image(1) generates OCR training pages. Given a text file it outputs an image with a given font and degradation.

OPTIONS

−−*text FILE*
File name of text input to use for creating synthetic training data. (type:string default:)

−−*outputbase FILE*
Basename for output image/box file (type:string default:)

−−*fontconfig_tmpdir PATH*
Overrides fontconfig default temporary dir (type:string default:/tmp)

−−*fonts_dir PATH*
If empty it use system default. Otherwise it overrides system default font location (type:string default:)

−−*font FONTNAME*
Font description name to use (type:string default:Arial)

−−*writing_mode MODE*
Specify one of the following writing modes. *horizontal* : Render regular horizontal text. (default) *vertical* : Render vertical text. Glyph orientation is selected by Pango. *vertical−upright* : Render vertical text. Glyph orientation is set to be upright. (type:string default:horizontal)

−−*tlog_level INT*
Minimum logging level for tlog() output (type:int default:0)

−−*max_pages INT*
Maximum number of pages to output (0=unlimited) (type:int default:0)

−−*degrade_image BOOL*
Degrade rendered image with speckle noise, dilation/erosion and rotation (type:bool default:true)

−−*rotate_image BOOL*
Rotate the image in a random way. (type:bool default:true)

−−*strip_unrenderable_words BOOL*
Remove unrenderable words from source text (type:bool default:true)

−−*ligatures BOOL*
Rebuild and render ligatures (type:bool default:false)

−−*exposure INT*
Exposure level in photocopier (type:int default:0)

−−*resolution INT*
Pixels per inch (type:int default:300)

−−*xsize INT*
Width of output image (type:int default:3600)

−−*ysize INT*
Height of output image (type:int default:4800)

−−*margin INT*
Margin round edges of image (type:int default:100)

−−*ptsize INT*
Size of printed text (type:int default:12)

−−*leading INT*
Inter−line space (in pixels) (type:int default:12)

−−*box_padding INT*
Padding around produced bounding boxes (type:int default:0)

−−*char_spacing DOUBLE*
Inter−character space in ems (type:double default:0)

−−*underline_start_prob DOUBLE*
Fraction of words to underline (value in [0,1]) (type:double default:0)

−−*underline_continuation_prob DOUBLE*
Fraction of words to underline (value in [0,1]) (type:double default:0)

−−*render_ngrams BOOL*
Put each space−separated entity from the input file into one bounding box. The ngrams in the input file will be randomly permuted before rendering (so that there is sufficient variety of characters on each line). (type:bool default:false)

−−*output_word_boxes BOOL*
Output word bounding boxes instead of character boxes. This is used for Cube training, and implied by −−render_ngrams. (type:bool default:false)

−−*unicharset_file FILE*
File with characters in the unicharset. If −−render_ngrams is true and −−unicharset_file is specified, ngrams with characters that are not in unicharset will be omitted (type:string default:)

−−*bidirectional_rotation BOOL*
Rotate the generated characters both ways. (type:bool default:false)

−−*only_extract_font_properties BOOL*
Assumes that the input file contains a list of ngrams. Renders each ngram, extracts spacing properties and records them in output_base/[font_name].fontinfo file. (type:bool default:false)

USE THESE FLAGS TO OUTPUT ZERO−PADDED, SQUARE INDIVIDUAL CHARACTER IMAGES

−−*output_individual_glyph_images BOOL*
If true also outputs individual character images (type:bool default:false)

−−*glyph_resized_size INT*
Each glyph is square with this side length in pixels (type:int default:0)

−−*glyph_num_border_pixels_to_pad INT*
Final_size=glyph_resized_size+2*glyph_num_border_pixels_to_pad (type:int default:0)

USE THESE FLAGS TO FIND FONTS THAT CAN RENDER A GIVEN TEXT

−−*find_fonts BOOL*
Search for all fonts that can render the text (type:bool default:false)

−−*render_per_font BOOL*
If find_fonts==true, render each font to its own image. Image filenames are of the form output_name.font_name.tif (type:bool default:true)

−−*min_coverage DOUBLE*
If find_fonts==true, the minimum coverage the font has of the characters in the text file to include it, between 0 and 1. (type:double default:1)

Example Usage: “‘ text2image −−find_fonts \ −−fonts_dir /usr/share/fonts \ −−text ../langdata/hin/hin.training_text \ −−min_coverage .9 \ −−render_per_font \ −−outputbase ../langdata/hin/hin \ |& grep raw | sed −e *s/ :. */″ \\/g* | sed −e *s/ˆ/ ″/ >*../langdata/hin/fontslist.txt “‘

SINGLE OPTIONS

−−list_available_fonts BOOL

List available fonts and quit. (type:bool default:false)

HISTORY

text2image(1) was first made available for tesseract 3.03.

RESOURCES

Main web site: **https://github.com/tesseract−ocr** Information on training tesseract LSTM: **https://github.com/tesseract−ocr/tesseract/wiki/TrainingTesseract−4.00**

SEE ALSO

tesseract(1)

COPYING

AUTHOR

The Tesseract OCR engine was written by Ray Smith and his research groups at Hewlett Packard (1985−1995) and Google (2006−present).