Gazette Extractor Documentation

Release 1.0.1

Maximiliana Behnke & Sandra Ambroziak

January 12, 2017

Contents

This is tutorial and documentation for Gazette Extractor - engineering project developed at Adam Mickiewicz University.

CHAPTER 1

Contents

1.1 Usage

1.1.1 Preparation

Firstly, user need to invoke *make install* command. It will install tools such as

- KenLM
- OpenCV for Python
- NLTK
- Vowpal Wabbit

After successful installation user need to prepare training set or testing set depending on usage. These sets should be put i

- training to *train* directory
- testing to *test-A* or *dev-0* directory

Directories should contain:

- newspapers in .djvu file format
- in.tsv files where newspapers titles and obituaries coordinates are kept

1.1.2 Train

To train vowpal wabbit model *make train -j X* command need to be invoked, where X is number of newspapers processed at the same time (train without flag -j means that just one newspaper will be processed at the same time).

During runtime the newspapers are being *unpacked*. The metadata, xml of page and its text are being extracted, the page is being saved into TIFF file. At the end every text layer of the newspaper pages are being merged into one text file which corresponds to text in whole newspaper. After unpacking the place takes *generate* process. During this step the rectangles are being generated basing on edges noticed by computer vision algorithms. When rectangles are generated the *classify* process gives them labels "-1" when the rectangle isn't an obituary and "1" when it is an obituary. Nextly LM and BPE model are being trained. In order to make LM corpora must be created:

- Corpus of necrologies

- Corpus of pages with necrologies

  BPE model is being trained on text layer of the newspaper extracted in *unpack* process. After training the *analyze* step is being running. During that process graphic and text features are being created and also language models are being queried. That features are necessary to create the input to vowpal wabbit model which is trained at the end of *train* process.

  Take a look into manual if you want to gain more information about invoking specific steps of training. It's useful when ones need to re-train model without unpacking all of the newspapers again.

1.1.3 Test

In this phase trained vowpal wabbit model is being tested. Just like in train point, user need to invoke *make test -j X* or *make dev -j X* command, where X is number of newspapers processed at the same time (test without flag -j means that just one newspaper will be processed at the same time).

Steps like unpacking and generating are same for both – test and train step. After that graphic, text and language models features of the newspaper are being created in analyze step. Files created during analyze are taken for prediction step – obituaries are being predicted in the newspapers.

Take a look into manual if you want to gain more information about invoking specific steps of testing. It's useful when ones need to re-test model without unpacking all of the newspapers again.

1.1.4 Cut necro

After testing - *out.tsv* file is created. It contains coordinates of the obituaries found in appropriate newspaper (from *in.tsv*). Command *make test-cut* or *make dev-cut* merges both of files and cut found obituary from the newspaper. The result can be found in *test-A/obituaries* or *dev-0/obituaries* directory.

1.1.5 Cleaning and purging

By clean is understood removing *out.tsv files, train files, corpora, arpa and klm files from LM directory, and BPE model. Purging removes also vw files and predict. Take a look into manual if you want to gain more information about invoking specific ways of purging and cleaning. formation about invoking specific ways of purging and cleaning.

1.1.6 Train flow

1.1.7 Test flow

1.2 Manual

1.2.1 Input

- **djvu file** of the newspaper in which user wants to find obituary.
- **in.tsv file (train)** : in this file are kept – filename of the newspaper and page number with coordinates where obituaries are placed format: newspaper_title.djvu page_number left_upper_x1_coordinate left_upper_y1_coordinate lower_right_x2_coordinate lower_right_y2_coordinate -> where separators are tabs.

- **in.tsv file (test)** : in this file filename of the newspaper is kept.

1.2.2 Install

- **make install** : install all of the requirements.
- **make install-doc** : install sphinx.

1.2.3 Train

- **make train-split** : creates .necro files (newspaper_title.necro) which include coordinates of obituary and page which contains that obituary.
- **make train** : trains vowpal wabbit model (launches : train-unpack, train-generate, train-lm, train-vw).
- **make train-unpack** : unpacks newspapers from train directory. Extracts metadata (title, type, language etc), page in tiff, xml and text layer of page.
- **make train-bpe** : trains BPE model based on txt files of newspapers.
- **make train-generate** : generates rectangles "noticed" on page – potential obituaries.
- **make train-classify** : tags generated rectangles using information stored in necro file.
- **make train-lm** : trains language models – char based 3-gram model of necrologies, BPE based 3-gram model of pages with necrologies
- **make train-analyze** : analyzes newspapers – extracts text and graphic features, counts language model score of page rectangles and language model score of page with current rectangle.
- **make train-merge** : merges vw files of newspapers into train.in file.
- **make train-vw** : trains vw model.
- **make train-purge** : removes necro, vw files and train.* from train directory; corpora, arpa and klm files from LM directory, BPE model from **BPE** directory.
- **make train-clean** : removes train.* files from train directory. corpora, arpa, klm files from LM directory, BPE model from *BPE* directory.

1.2.4 Test

- **make test** : tests trained vw model.
- **make test-unpack**: unpacks newspapers from *test-A* directory. Extracts metadata (title, type, language etc), page in tiff, xml and text layer of page.
- **make test-generate** : generates rectangles of page – potential obituaries.
- **make test-analyze** : analyzes newspapers – extracts text and graphic features, counts language model score of page rectangles and language model score of page with current rectangle.
- **make test-predict** : predicts obituaries for all newspapers in *test-A* directory.
- **make test-merge** : creates out.tsv file where coordinates of obituaries are kept.
- **make test-purge** : removes vw, predict, out.tsv and newspaper_title.out.tsv files from test directory.
- **make test-clean** : removes out.tsv and newspaper_title.out.tsv files from test directory.

1.2.5 Dev

- **make dev** : tests trained vw model.
- **make dev-unpack**: unpacks newspapers from *dev-0* directory. Extracts metadata (title, type, language etc), page in tiff, xml and text layer of page.
- **make dev-generate** : generates rectangles of page – potential obituaries.
- **make dev-analyze** : analyzes newspapers – extracts text and graphic features, counts language model score of page rectangles and language model score of page with current rectangle.
- **make dev-predict** : predicts obituaries for all newspapers in *dev-0* directory.
- **make dev-merge** : creates out.tsv file where coordinates of obituaries are kept.
- **make dev-purge** : removes vw, predict, out.tsv and newspaper_title.out.tsv files from *dev-0* directory.
- **make dev-clean** : removes out.tsv and newspaper_title.out.tsv files from *dev-0* directory.

1.2.6 Clean

- **make clean-unpack** : removes txt files from train directory and files created by unpack command.
- **make clean-generate** : removes files created by generate command.
- **make clean-classify** : removes files created by classify command.
- **make clean-analyze** : removes files created by analyze command.
- **make purge** : removes vw, predict, out.tsv and newspaper_title.out.tsv files from test directory; necro, vw files and train.* from train directory; corpora, arpa and klm files from LM directory, BPE model from *BPE* directory.
- **make clean** : removes out.tsv and newspaper_title.out.tsv files from test directory; train.* files from train directory; corpora, arpa, klm files from LM directory, BPE model from *BPE* directory.

1.2.7 Doc

- **make -f doc_maker html** : creates automatically generated documentation of python modules.
- **make -f doc_maker clean** : removes content of build directory.

1.2.8 Cut

- **make dev-cut** : cuts obituary from the newspaper, bases on merged out.tsv and in.tsv files, puts obutuaries into *dev-0/obituaries* directory.
- **make test-cut** : cuts obituary from the newspaper, bases on merged out.tsv and in.tsv files, puts obutuaries into *test-A/obituaries* directory.

1.3 scripts

1.3.1 classify_rectangles module

class `classify_rectangles.`**`Classifier`**(*r_file*, *n_file*, *page*, *error*)
Classify each rectangle if it's obituary or not based on training data.

Note: If Classifier does not find obituary, it adds it to already generated rectangles.

check_error(*necrology*, *rectangle*)
Check coordinates of rectangle error (how near it is to necrology). Return error value of all 4 nodes.

Args: necrology (tuple): coordinates of compared necrology. rectangle (tuple): coordinates of compared rectangle.

classify(*page*, *error*)
Tag all rectangles with classes based on necrologies' coordinates.

Args: i (int): Page number.

load_necrologies(*n_file*)
Load necrologies' coordinates from file.

Args: n_file (str): Path to file with necrologies' coordinates.

load_rectangles(*r_file*)
Loads rectangles' coordinates from file.

Args: r_file (str): Path to file with rectangles' coordinates.

modify_rectangle_file(*r_file*, *modified*)
If any necrology is not found in rectangles, rectangle file is modified.

Args: modified (bool): flag that checks if rectangle file should be modified.

classify_rectangles.**parse_args**()
Parse command line arguments.

1.3.2 common_text_features_functions module

Module which contains functions which are used more than once in text features extraction scripts

common_text_features_functions.**cut_xml**(*_x1*, *_y1*, *_x2*, *_y2*, *xml_file*)
Returns text, contained in specified area (recognized rectangle in newspaper), from xml file.

Args: _x1 (int) : Upper left-sided x coordinate _y1 (int) : Upper left-sided y coordinate _x2 (int) : Lower right-sided x coordinate _y2 (int) : Lower right-sided y coordinate

common_text_features_functions.**get_punct_amount**(*words_list*)
Returns number of punctation in desired rectangle.

Args: words_list (list) : list of words in which we need to check amount of punctation

1.3.3 create_corpus_necrologies module

Script which creates a corpus based on nercrologies. Normalization - lowercasing.

1.3.4 create_corpus_pages module

Script which creates a corpora based on pages which contain necrologies. Normalization - lowercasing.

1.3.5 cut_necro module

1.3.6 detect_peaks module

1.3.7 extract_necro module

extract_necro.**extract_necro**()

extract_necro.**parse_vector**(*line*)

1.3.8 graphic_features_extractor module

1.3.9 lm_feature module

1.3.10 merge_necro module

Script to map extracted obituaries in order of DJVU files.

merge_necro.**load_input**()
Load extracted necrologies from stdin.

merge_necro.**map_output**(*i_file*)
Map extracted necrologies by filenames order in.tsv file.

Args: i_file (str): Path to in.tsv file.

merge_necro.**parse_args**()
Parse command line arguments.

1.3.11 metadata_extract module

1.3.12 rectangle module

1.3.13 split_necro module

Split training data into seperate files (each for DDJVU file).

1.3.14 text_features_extractor module

1.3.15 xml_cleaner module

Removes forbidden chars from xml file. By forbidden chars are understood : - chars which cause xml parser failing. - non unicode chars.

1.3.16 xml_extract module

Prints coordinates of paragraphs, words and lines due to given .xml file .xml file needs to be cleaned by xml_cleaner.py Helps in checking what is word and what is the picture fragment.

xml_extract.**check_paragraph**(*para_xml*)
Checks if paragraphs contains trash, returns true if not and false if yes

Args: para_xml (str) : xml of paragraph

xml_extract.**create_output**()
Prints out the final output

xml_extract.**create_words_lines_output**(*coordinates_words*)
Function which helps in making data for lines and words

xml_extract.**get_alpha**(*line*)
Returns amount of alphanumeric chars.

Args: line (str) : string which needs to be checked

xml_extract.**get_lines_xml**(*para_xml*)
Get lines from xml

Args: para_xml (str) : xml of "PARAGRAPH"

xml_extract.**get_paragraphs_xml**(*root*)
Get paragraphs from xml

xml_extract.**get_words_xml**(*line_xml*)
Get words from xml file

Args: line_xml (str) : xml of "LINE"

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

c

e

m

s

x

C

E

G

L

M

P

S

X