Article Quality Documentation

Release 0.4.4

Aaron Halfaker & Morten Warncke-Wang

Dec 21, 2022

Contents

A library for performing automatic detection of assessment classes of Wikipedia articles.

- **Install:** `pip install articlequality`
- **Models:** https://github.com/wikimedia/articlequality/tree/master/models
- **Repo:** https://github.com/wikimedia/articlequality
- **License:** MIT License

CHAPTER 1

Contents

1.1 Utilities

This module implements a set of utilities for extracting labeling events, text and features from the command-line. When the articlequality python package is installed, a *articlequality* utility should be available from the commandline. Run *revscoring -h* for more information:

1.1.1 Article Quality CLI

articlequality

```
$ articlequality -h
```

```
This script provides access to a set of utilities for extracting features
and building article quality classifiers.

* extract_labelings -- Gathers quality labeling events from XML dumps
* extract_text -- Gathers text for each labeling observation from XML dumps
* extract_features -- Extracts feature_lists for observations
* fetch_item_info -- Gets interesting statements from wikidata items
* fetch_text -- Gathers text for each labeling observation from a MediaWiki
                API

Usage:
    articlequality (-h | --help)
    articlequality <utility> [-h | --help]

Options:
    -h | --help  Prints this documentation
    <utility>    The name of the utility to run
```

1.1.2 Sub-utilities

extract_from_text

```
$ articlequality extract_from_text -h
```

```
Extracts dependents from a labeling doc containing text and an
`wp10` label writes a new set of labeling docs that is
compatible as observations for `revscoring`'s cv_train and tune utilities.

Input: { ... "wp10": ..., "text": ..., ... }

Output: { ... "wp10": ..., "cache": ..., ... }


Usage:
    extract_from_text <dependent>...
                      [--input=<path>]
                      [--output=<path>]
                      [--extractors=<num>]
                      [--verbose]
                      [--debug]

Options:
    -h --help               Print this documentation
    <dependent>             Classpath to a single dependent or list of
                            dependent values to solve
    --input=<path>          Path to a file containing observations
                            [default: <stdin>]
    --output=<path>         Path to a file to write new observations to
                            [default: <stdout>]
    --extractors=<num>      The number of parallel extractors to
                            start [default: <cpu count>]
    --verbose               Print dots and stuff to stderr
    --debug                 Print debug logs
```

extract_labelings

```
$ articlequality extract_labelings -h
```

```
Extracts labels from an XML dump and writes out labeled observations for
each change in assessment class.  Will match extraction method to the dump.

Usage:
    extract_labelings <dump-file>... [--extractor=<name>] [--threads=<num>]
                                     [--output=<path>] [--verbose]
                                     [--debug]
    extract_labelings -h | --help

Options:
    -h --help           Show this screen.
    <dump-file>         An XML dump file to process
    --extractor=<name>  The dbname of the wiki extractor to use
                        (e.g. 'enwiki')  [default: <match>]
    --threads=<num>     If a collection of files are provided, how many
                        processor threads should be prepare?
```

(continues on next page)

(continued from previous page)

```
                          [default: <cpu_count>]
    --output=<path>       The path to a file to dump observations to
                          [default: <stdout>]
    --verbose             Prints dots to <stderr>
    --debug             Print debug level logging
```

extract_text

```
$ articlequality extract_text -h
```

```
Extracts text & metadata for labelings using XML dumps.

Usage:
    extract_text <dump-file>... [--labelings=<path>] [--output=<path>]
                                [--threads=<num>] [--verbose]
    extract_text -h | --help

Options:
    -h --help             Show this screen.
    <dump-file>           An XML dump file to process
    --labelings=<name>    The path to a file containing labeling events.
                          [default: <stdin>]
    --output=<path>       The path to a file to dump observations to
                          [default: <stdout>]
    --threads=<num>       If a collection of files are provided, how many
                          processor threads should be prepare?
                          [default: <cpu_count>]
    --verbose             Prints dots to <stderr>
```

fetch_text

```
$ articlequality fetch_text -h
```

```
Fetches text & metadata for labelings using a MediaWiki API.

Usage:
    fetch_text --api-host=<url> [--labelings=<path>] [--output=<path>]
                                [--verbose]

Options:
    -h --help             Show this documentation.
    --api-host=<url>      The hostname of a MediaWiki e.g.
                          "https://en.wikipedia.org"
    --labelings=<path>    Path to a containting observations with extracted
                          labels. [default: <stdin>]
    --output=<path>       Path to a file to write new observations
                          (with text) out to. [default: <stdout>]
    --verbose             Prints dots and stuff to stderr
```

score

```
$ articlequality score -h
```

```
Applies a scoring model to a chunch of text.

Usage:
    score <model-file> [<text>]
    score -h | --help

Options:
    -h --help     Prints this documentation
    <model-file>  The path to a scorer_model file to use
    <text>        The path to a file containing text to score
                  [default: <stdin>]
```

1.2 Extractors

This module provides a set of *articlequality.Extractor* s that implement a strategy for identifying article quality labeling events historically. These labelings are used as training data to build prediction models.

1.2.1 Supported wikis

1.2.2 Base classes

class articlequality.**Extractor**(*name*, *doc*, *namespaces*)

Implements an labeling event extraction strategy.

Parameters

name [*str*] A name for the extraction strategy

doc [*str*] Documentation describing the extraction strategy

namespace [*iterable‘(‘int*)] A set of namespaces that will be considered when performing an extraction

extract(*page*, *verbose=False*)

Processes an mwxml.Page and returns a generator of first-observations of a project/label pair.

Parameters

page [mwxml.Page] Page to process

verbose [*bool*] print dots to stderr

invert_reverted_status(*reverteds*, *revisions*)

This method recursively searches the reverted status of revisions and inverts the status when reverts are themselves reverted.

class articlequality.**TemplateExtractor**(**args*, *from_template*, ***kwargs*)

Implements a template-based extraction strategy based on a *from_template* function that takes a template and returns a (project, label) pair.

Parameters

from_template [*func*] A function that takes a template and returns a (project, label) pair

extract(*page*, *verbose=False*)

Processes an mwxml.Page and returns a generator of first-observations of a project/label pair.

Parameters

page [mwxml.Page] Page to process

verbose [*bool*] print dots to stderr

extract_labels(*text*)

Extracts a set of labels for a version of text by parsing templates.

Parameters

text [*str*] Wikitext markup to extract labels from

Returns An iterator over (project, label) pairs

invert_reverted_status(*reverteds*, *revisions*)

This method recursively searches the reverted status of revisions and inverts the status when reverts are themselves reverted.

CHAPTER 2

Basic usage

```
>>> import articlequality
>>> from revscoring import Model
>>>
>>> scorer_model = Model.load(open("models/enwiki.wp10.rf.model", "rb"))
>>>
>>> text = "I am the text of a page.  I have a <ref>word</ref>"
>>> articlequality.score(scorer_model, text)
{'prediction': 'stub',
 'probability': {'stub': 0.27156163795807853,
                 'b': 0.14707452309674252,
                 'fa': 0.16844898943510833,
                 'c': 0.057668704007171959,
                 'ga': 0.21617801281707663,
                 'start': 0.13906813268582238}}
```

CHAPTER 3

Authors

- Aaron Halfaker – https://github.com/halfak
- Morten Warncke-Wang – https://github.com/nettrom

```
MIT LICENSE

Copyright (c) 2015 Aaron Halfaker <ahalfaker@wikimedia.org>

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom
the Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall
be included in all copies or substantial portions of the
Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS
OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index