Segmentation Evaluation Documentation

Release 2.0.11

Chris Fournier

May 13, 2017
Contents

1 Feature Support 3

2 User Guide 5
  2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  2.2 Installation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
  2.3 Quickstart . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

3 API Documentation 11
  3.1 Developer Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

4 Support 19

5 Citing SegEval 21

6 References 23

Bibliography 25

Python Module Index 27
A package providing text segmentation evaluation metrics and utilities. (Installation)

Text segmentation is the task of splitting up any amount of text into segments by placing boundaries between some atomic unit (e.g., morphemes, words, lines, sentences, paragraphs, sections, etc.). It’s a common pre-processing step in many Natural Language Processing (NLP) tasks.

E.g., if we were to perform both manual and automatic syllabification of words, one may need a way to compare how close the automatic solution is to the manual one. For this, we can use Boundary Edit Distance and Boundary Similarity. Evaluating a hypothetical automatic syllabifier, we can obtain the results shown below.

<table>
<thead>
<tr>
<th>Word</th>
<th>Manual Solution</th>
<th>Automatic Solution</th>
<th>Boundary Edit Distance</th>
<th>Boundary Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>automatic</td>
<td>au-to-ma-tic</td>
<td>au-tom-a-tic</td>
<td>2 matches, 1 near</td>
<td>0.83</td>
</tr>
<tr>
<td>segmentation</td>
<td>seg-men-ta-tion</td>
<td>seg-ment-ation</td>
<td>1 match, 1 near, 1 miss</td>
<td>0.50</td>
</tr>
<tr>
<td>is</td>
<td>is</td>
<td>is</td>
<td>No edits</td>
<td>1.00</td>
</tr>
<tr>
<td>fun</td>
<td>fun</td>
<td>f·un</td>
<td>1 miss</td>
<td>0.00</td>
</tr>
</tbody>
</table>

This package is a collection of metrics and for comparing text segmentations and evaluating automatic text segmenters. Both new (Boundary Similarity, Segmentation Similarity) and traditional (WindowDiff, Pk) are included, as well as inter-coder agreement coefficients and confusion matrices based upon a boundary edit distance.

For more examples of how to use SegEval, see “An initial study of topical poetry segmentation”.

Release 2.0.11 (changelog)

Date May 13, 2017
A variety of segmentation comparison metrics are implemented, including:

- Boundary Edit Distance (BED; [Fournier2013])
- Boundary Similarity (B; [Fournier2013])
- BED-based confusion matrices (and precision/recall/F1; [Fournier2013])
- Segmentation Similarity (S; [FournierInkpen2012])
- WindowDiff [PevznerHearst2002]
- Pk [BeefermanBerger1999]

Additionally, B-based inter-coder agreement coefficients for segmentation that are suitable for 2 or more coders are provided, including:

- Fleiss’ $\pi$ [Fleiss1971] (i.e., Siegel and Castellan’s $K$ [SiegelCastellan1988])
- Fleiss’ $\kappa$ [DaviesFleiss1982]
This part of the documentation, which is mostly prose, begins with some background information about Requests, then focuses on step-by-step instructions for getting the most out of Requests.

**Introduction**

This package aims to make comparing text segmentations and evaluating one's segmentation data and methods easier. It implements the new segmentation comparison metrics detailed in [Fournier2013] and [Fournier2013b].

**License**

Copyright (c) 2011-2013 Chris Fournier

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the author nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
Installation

This part of the documentation covers the installation of SegEval. The first step to using any software package is getting it properly installed.

Distribute & Pip

Installing segeval can be performed using pip:

$ pip install segeval

Get the Code

Segeval’s code is available on GitHub.

You can either clone the public repository:

git clone git://github.com/cfournie/segmentation.evaluation.git

Download the tarball:

$ curl -OL https://github.com/cfournie/segmentation.evaluation/tarball/master

Or, download the zipball:

$ curl -OL https://github.com/cfournie/segmentation.evaluation/zipball/master

Once you have a copy of the source, you can embed it in your Python package, or install it into your site-packages easily:

$ python setup.py install

Quickstart

This page gives a good introduction in how to get started with SegEval. This assumes you already have SegEval installed. If you do not, head over to the Installation section.

Let’s get started with some simple examples.

Loading Data

Start by encoding the size of each segment produced by each coder for a document (e.g. the Stargazer’s text segmented in [Hearst1997]) as JSON using the format shown below.
Begin by importing the SegEval module:

```python
>>> import segeval
```

Now, let's import this data using the `input_linear_mass_json()` function:

```python
>>> dataset = segeval.input_linear_mass_json('hearst1997.json')
```

Now, we have a `Dataset` object called `dataset`. We can compute a variety of statistics upon this data.

### Comparing Segmentations

Given a `dataset`, if we wanted to compare two coder’s responses together, we can select the coder’s that we care about much like how one accesses arrays/dictionary items:

```python
>>> import segeval
>>> dataset = segeval.HEARST_1997_STARGAZER
>>> segmentation1 = dataset['stargazer'][1]
>>> segmentation2 = dataset['stargazer'][2]
```

Segmentations can then be compared using functions such as:

```python
>>> segeval.boundary_similarity(segmentation1, segmentation2)
Decimal('0.5')
```

Other metrics are also available, including:

```python
>>> segeval.segmentation_similarity(segmentation1, segmentation2)
Decimal('0.825')
```

If instead of one metric you desire a large number of statistics about the difference between two boundaries, you can use:

```python
>>> segeval.boundary_statistics(segmentation1, segmentation2)
```

This produces:

```json
{
    'matches': [1, 1, 1], # List of matching boundary types
    'boundaries_all': 11,
    'pbs': 20, # Potential boundaries
}
```
If instead we had a hypothetical segmentation generated by an automatic segmenter and we wanted to compare it against `segmentation1`, we could use these metrics:

```python
>>> hypothesis = (2, 6, 4, 2, 4, 3)
>>> reference = dataset['stargazer'][:1]
>>> segeval.boundary_similarity(hypothesis, reference)
Decimal('0.5714285714285714285714285714')
```

Some traditional segmentation comparison metrics can also be used:

```python
>>> segeval.window_diff(hypothesis, reference)
Decimal('0.3157894736842105263157894737')
>>> segeval.pk(hypothesis, reference)
Decimal('0.2631578947368421052631578947')
```

If instead one wants to analyze this as a boundary classification task, we can produce a confusion matrix using:

```python
>>> confusion_matrix = segeval.boundary_confusion_matrix(hypothesis, reference)
```

This produces a `ConfusionMatrix` object named `confusion_matrix`. This confusion matrix can then be passed to information retrieval metrics, such as:

```python
>>> segeval.precision(confusion_matrix)
Decimal('0.5714285714285714285714285714')
>>> segeval.recall(confusion_matrix)
Decimal('0.5714285714285714285714285714')
>>> segeval.fmeasure(confusion_matrix)
Decimal('0.72727272727272727272727267')
```

All of these functions can be used on either pairs of segmentations, single `Dataset` objects (computing pairwise values), and two `Dataset` objects (comparing the coders in one to all coders in another). Comparing two `Dataset` objects is how one could compare a set of automatic segmenters to a set of human segmenters to evaluate the performance of the automatic segmenters, for example:

```python
>>> manual = segeval.HEARST_1997_STARGAZER
>>> automatic = segeval.HYPOTHESIS_STARGAZER
>>> segeval.boundary_similarity(manual, automatic)
```

This produces:

```
{'stargazer,3,h2': Decimal('0.5'),
 'stargazer,3,h1': Decimal('0.45'),
 'stargazer,6,h1': Decimal('0.5833333333333333333333333333'),
 'stargazer,1,h1': Decimal('0.5714285714285714285714285714'),
 'stargazer,1,h2': Decimal('0.3888888888888888888888888889'),
 'stargazer,6,h2': Decimal('0.3888888888888888888888888889'),
 'stargazer,7,h2': Decimal('0.3181818181818181818181818182'),
 'stargazer,7,h1': Decimal('0.5')},
```
Note that the key for each value is the document name (stargazer), followed by the coder from the manual dataset (e.g., 3) and the coder from the automatic dataset (e.g., h2).

### Computing Inter-Coder Agreement

Given a dataset, if we wanted to compute the actual agreement between all coders using `boundary_similarity()` we can use:

```python
>>> import segeval
>>> dataset = segeval.HEARST_1997_STARGAZER
>>> segeval.actual_agreement_linear(dataset)
Decimal('0.5300546448087431693989071038')
```

If instead one would like to use `segmentation_similarity()`, we can specify this function:

```python
>>> segeval.actual_agreement_linear(dataset, fnc_compare=segeval.segmentation_similarity)
Decimal('0.7952380952380952380952380952')
```

If instead we want a chance-corrected inter-coder agreement coefficient, Fleiss’ $\kappa$ and $\pi$ adapted to use `boundary_similarity()` can be used:

```python
>>> segeval.fleiss_kappa_linear(dataset)
Decimal('0.4414910889068254984367317023')
>>> segeval.fleiss_pi_linear(dataset)
Decimal('0.4405412438199323445225084569')
```
If you are looking for information on a specific function, class or method, this part of the documentation is for you.

**Developer Interface**

The APIs for most metrics can be provided either two segmentations to compare or a dataset to perform pairwise comparisons upon. There are a variety of parameters that can be specified other than that which is compared, but all have defaults specified.

**Boundary-Edit-Distance-based Metrics**

These segmentation comparison metrics were introduced in [Fournier2013].

```python
segeval.boundary_statistics(*args, **kwargs)
```

Computes a large number of BED-based and other segmentation statistics, returning a `dict()` that includes:

- `count_edits`, a count of BED edits;
- `additions`, a list of BED addition edits;
- `substitutions`, a list of BED substitution edits;
- `transpositions`, a list of BED transposition edits;
- `full_misses`, a list of fully-missed boundaries (regardless of edits);
- `boundaries_all`, a count of boundaries compared;
- `matches`, a list of matching boundaries;
- `pbs`, a count of potential boundary types.

```python
class segeval.BoundaryFormat
```

An `enum` with options that include:
• sets, a boundary set string; see `boundary_string_from_masses()`
• mass, a tuple of segment masses; see `convert_positions_to_masses()`
• position, a tuple of position segment labels; see `convert_masses_to_positions()`
• nltk, a string representation of segment positions; see `convert_nltk_to_masses()`

**Boundary Similarity (B)**

This metric compares the correctness of boundary pairs between segmentations [Fournier2013].

**Note:** This is a recommended segmentation comparison metric for situations when there is no reference segmentation to compare against; see [Fournier2013].

```python
segeval.boundary_similarity(segmentation_a, segmentation_b, **kwargs)
```

**Parameters**
- `segmentation_*` (segmentation or Dataset) – Segmentation or dataset containing segmentations of a particular format; see `BoundaryFormat`

```python
segeval.boundary_similarity(dataset, **kwargs)
```

**Parameters**
- `dataset` (Dataset) – Dataset of segmentations

```python
segeval.boundary_similarity()
```

**Parameters**
- `boundary_format` (BoundaryFormat enum) – Segmentation format; default `BoundaryFormat.mass`
- `permuted` (bool) – Use pairwise permutations v.s. combinations; default False
- `one_minus` (bool) – Return $1 - value$; default False
- `return_parts` (bool) – Return tuples of numerators, denominators, or other values comprising a metric; default False
- `n_t` (int) – See `boundary_edit_distance()`
- `boundary_types` (set) – Set of allowable boundary types; default set([1])
- `weight` (tuple) – Tuple of weighting functions, see Weighting Functions; default is scaling of substitution and transposition but not addition edits (`weight_a()`, `weight_s_scale()`, `weight_t_scale()`)

**Segmentation Similarity (S)**

Originally introduced in [FournierInkpen2012], this metric uses the revised boundary edit distance in [Fournier2013] and compares segmentations to provide the proportion of unedited potential boundary positions.

**Warning:** Prefer `boundary_similarity()` instead; see [Fournier2013].

```python
segeval.segmentation_similarity(segmentation_a, segmentation_b, **kwargs)
```

For parameters see `boundary_similarity()`

```python
segeval.segmentation_similarity(dataset, **kwargs)
```

For parameters see `boundary_similarity()`
Boundary Edit Distance (BED)

An edit distance proposed in [Fournier2013] that operates upon boundaries to produce:

- Additions/deletion edits to model full misses,
- Transposition edits to model near misses, and
- Substitution edits to model boundary-type confusion.

For more details, see Section 3.1 of [Fournier2013b].

Computes boundary edit distance between two boundary strings. Returns a list of Addition, Substitution, and Transposition edit sets.

Parameters

- `boundary_string_a` (tuple) – Boundary string to compare; produced by `boundary_string_from_masses()`
- `boundary_string_b` (tuple) – See `boundary_string_a`
- `n_t` (int) – Maximum distance (in potential boundary positions) that a transposition may span

BED-based Confusion Matrix (BED-CM)

A confusion-matrix-formulation proposed in [Fournier2013] that uses BED to populate a matrix by using matches and scaled transpositions as correct classifications for boundary types, substitutions as confusion between boundary types, and additions/deletions as missing boundary types.

Note: This is a recommended segmentation comparison metric, when summarized by an information-retrieval metric such as `precision()`, `recall()`, `fmeasure()`, etc., for situations when there is a reference segmentation to compare against; see [Fournier2013].

Weighting Functions

These functions are used by BED-based metrics to weight edit operations.

Default unweighted weighting function for addition edit operations.
segeval.weight_s(substitutions, max_s, min_s=1)
Unweighted weighting function for substitution edit operations.

segeval.weight_s_scale(substitutions, max_s, min_s=1)
Default weighting function for substitution edit operations by the distance between ordinal boundary types.

segeval.weight_t(transpositions, max_n)
Unweighted weighting function for transposition edit operations.

segeval.weight_t_scale(transpositions, max_n)
Default weighting function for transposition edit operations by the distance that transpositions span.

Traditional Metrics

segeval.compute_window_size(reference, **kwargs)

Pk

Proposed in [BeefermanBerger1999], this segmentation comparison metric runs a window over a hypothesis and reference segmentation and counts those hypothesis windows whose ends are in differing segmentations that do not agree with the reference window as being in error. These errors are then summed over all windows.

Warning: Prefer boundary_similarity() instead; see [Fournier2013].

segeval.pk(hypothesis, reference, **kwargs)

Parameters

• hypothesis (segmentation or Dataset) – Hypothetical, or automatically-generated, segmentation (or dataset of segmentations) of a particular format; see BoundaryFormat

• reference (segmentation or Dataset) – Reference, or manually-created, segmentation (or dataset of segmentations) of a particular format; see BoundaryFormat

segeval.pk(dataset, **kwargs)

Parameters dataset (Dataset) – Dataset of segmentations

segeval.pk()

Parameters

• boundary_format (BoundaryFormat enum) – Segmentation format; default BoundaryFormat.mass

• permuted (bool) – Use pairwise permutations v.s. combinations; default True

• one_minus (bool) – Return 1 − value; default False

• return_parts (bool) – Return tuples of numerators, denominators, or other values comprising a metric; default False

• window_size (int) – Overriding window size – if not None, this replaces the per-comparison window size computed using compute_window_size() as the window size used; default None

• fnc_round (function) – Rounding function used when computing window size, see compute_window_size(); default round()
WindowDiff

Proposed in [PevznerHearst2002], this segmentation comparison metric is an adaptation of Pk which runs a window over a hypothesis and reference segmentation and counts those hypothesis windows with differing numbers of contained boundaries that do not agree with the reference window as being in error. These errors are then summed over all windows.

**Warning:** Prefer `boundary_similarity()` instead; see [Fournier2013].

```python
segeval.window_diff(hypothesis, reference, **kwargs)
For parameters see `pk()`
```

```python
segeval.window_diff(dataset, **kwargs)
For parameters see `pk()`
```

```python
segeval.window_diff()
For parameters see `pk()`
```

Inter-coder Agreement Coefficients

Originally adapted in [FournierInkpen2012] from formulations provided by [ArtsteinPoesio2008], these have inter-coder agreement have been modified by [Fournier2013] to better suite the measurement of inter-coder agreement of segmentation boundaries using `boundary_similarity()` for actual agreement.

```python
segeval.actual_agreement_linear()
Calculate actual (i.e., observed or textA_a), boundary agreement without accounting for chance, using [ArtsteinPoesio2008]'s formulation as adapted by [Fournier2013].
```

**Parameters**

- `fnc_compare` (*function*) – Segmentation comparison metric function to use; default `boundary_similarity()`
- `boundary_format` (*BoundaryFormat* enum) – Segmentation format; default `BoundaryFormat.mass`
- `permuted` (*bool*) – Use pairwise permutations v.s. combinations; default `False`
- `one_minus` (*bool*) – Return `1 - value`; default `False`
- `return_parts` (*bool*) – Return tuples of numerators, demoninators, or other values comprising a metric; default `False`
- `n_t` (*int*) – See `boundary_edit_distance()`
- `boundary_types` (*set*) – Set of allowewable boundary types; default `set([1])`
- `weight` (*tuple*) – Tuple of weighting functions, see Weighting Functions; default is scaling of substitution and transposition but not addition edits (`weight_a()`, `weight_s_scale()`, `weight_t_scale()`)  

```python
segeval.fleiss_pi_linear(dataset, **kwargs)
Calculates Fleiss’ π (or multi-π), originally proposed in [Fleiss1971], and is equivalent to Siegel and Castellan’s K [SiegelCastellan1988]. For 2 coders, this is equivalent to Scott’s π [Scott1955].
```

For parameters see `actual_agreement_linear()`
segeval.fleiss_kappa_linear(dataset, **kwargs)
Calculates Fleiss' $\kappa$ (or multi-$\kappa$), originally proposed in [DaviesFleiss1982]. For 2 coders, this is equivalent to Cohen’s $\kappa$ [Cohen1960].
For parameters see actual_agreement_linear()

segeval.artstein_poesio_bias_linear(dataset, **kwargs)
Artstein and Poesio’s annotator bias [ArtsteinPoesio2008].
For parameters see actual_agreement_linear()

Format Conversion

These utility functions are used internally and provided to allow for the conversion between the supported segmentation formats (see BoundaryFormat).

segeval.boundary_string_from_masses(masses)
Creates a “boundary string”, or sequence of boundary type sets from a list of segment masses, e.g., [5, 3, 5] becomes [(), (), (), (), (), (), (), (), (), ()].

Parameters masses (tuple) – Segmentation masses.

segeval.convert_positions_to_masses(positions)
Convert an ordered sequence of boundary position labels into a sequence of segment masses, e.g., [1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3] becomes [5, 3, 5].

Parameters segments (tuple) – Ordered sequence of which segments a unit belongs to.
Depreced since version 1.0.

segeval.convert_masses_to_positions(masses)
Converts a sequence of segment masses into an ordered sequence of section labels for each unit, e.g., [5, 3, 5] becomes [1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3].

Parameters masses (tuple) – Segment mass sequence.

segeval.convert_nltk_to_masses(string, boundary_symbol='1')
Convert an NLTK-formatted segmentation into masses, e.g., 000001000100000 becomes [5, 3, 5].
For more information, see nltk.metrics.segmentation.

Parameters

• string (str) – NLTK-formatted segmentation.
• boundary_symbol (str) – String that represents a boundary.

Data

These classes and functions deal with segmentation data representation and manipulation.

Model

These classes are used to model and store text (i.e., item) segmentations (i.e., codings).

class segeval.Dataset (item_coder_data=None, properties=None, boundary_types=None, boundary_format='mass')
Represents a set of texts (i.e., items) that have been segmentations by coders.
copy()
Create a deep copy of the entire dataset object and properties.

class segeval.Field
An enum with options representing json fields when storing segmentations which include:
• segmentation_type, the type if segmentation; default is SegmentationType.linear
• items, items with annotators and codings stored within
• codings, annotators and codings stored within

class segeval.SegmentationType
An enum with options representing segmentation structure types including:
• linear, linear segmentation

Input/Output

These functions serialization and de-serialization segmentation datasets. The recommended serialization format is JSON.

See also:

JSON (JavaScript Object Notation)

segeval.input_linear_mass_tsv(filepath, delimiter='\t')
Takes a file path. Returns segmentation mass codings as a Dataset.

Parameters
• filepath (str) – path to the mass file containing segment mass codings.
• delimiter (str) – the delimiter used when reading a TSV file (by default, a tab, but it can also be a comma, whitespace, etc.

segeval.input_linear_mass_json(filepath)
Reads a file path. Returns segmentation mass codings as a Dataset.

Parameters filepath (str) – Path to the mass file containing segment position codings.

segeval.output_linear_mass_json(filepath, dataset)
Takes a file path and Dataset and serializes it as JSON.

Parameters filepath (str) – Path to the mass file containing segment position codings.

segeval.load_nested_folders_dict(containing_dir, filetype=None, dataset=None, prepend_item=None)
Loads TSV files from a file directory structure, which reflects the directory structure in nested dict() with each directory name representing a key in these dict().

Parameters
• containing_dir (str) – Root directory containing sub-directories which contain segmentation files.
• filetype (str) – File type to load (e.g., json or tsv).

Information-Retrieval-related Statistics

segeval.precision(matrix, classification=None, version=0)
Calculate precision.

3.1. Developer Interface
Parameters

- `matrix (ConfusionMatrix)` – Confusion matrix
- `classification` (Any dict index) – Classification label to compute this metric for
- `version` (Average) – Averaging-method version.

`segeval.recall(matrix, classification=None, version=0)`
Calculate recall.

Parameters

- `matrix (ConfusionMatrix)` – Confusion matrix
- `classification` (Any dict index) – Classification label to compute this metric for
- `version` (Average) – Averaging-method version.

`segeval.fmeasure(matrix, classification=None, beta=Decimal('1.0'), version=0)`
Calculate FMeasure.

Parameters

- `matrix (ConfusionMatrix)` – Confusion matrix
- `classification` (Any dict index) – Classification label to compute this metric for
- `version` (Average) – Averaging-method version.

`segeval.summarize(pairs)`
Takes a list of values and returns the mean, standard deviation, variance, standard error, and number of values.

Parameters `pairs (list)` – List of numerical values

Model

Classes used to model segmentation comparisons so that they can be summarized by information retrieval related statistics (e.g., precision()).

```python
class segeval.Average

An enum with options representing the methods of computing averages:

- `micro`, micro-average
- `macro`, macro-average

For more details, see the Stanford IR Book.
```

```python
class segeval.ConfusionMatrix

A dict()-like representation of a confusion matrix offering some automation. To access/store values, use:

```
matrix[predicted][actual].
```

```python
classes()

Retrieves the set of all classes.
```
If you have any suggestions, problems, or difficulties, please log an issue, or contact me.
Citing SegEval

If you’re using this software for research, please cite the ACL paper [Fournier2013] and, if you need to go into details, the thesis [Fournier2013b] describing this work.

BibTeX:

```latex
@inproceedings{Fournier2013a,
  author    = {Fournier, Chris},
  year      = {2013},
  title     = {{Evaluating Text Segmentation using Boundary Edit Distance}},
  booktitle = {Proceedings of 51st Annual Meeting of the Association for Computational Linguistics},
  publisher = {Association for Computational Linguistics},
  location  = {Sophia, Bulgaria},
  pages     = {to appear},
  address   = {Stroudsburg, PA, USA}
}

@mastersthesis{Fournier2013b,
  author    = {Fournier, Chris},
  title     = {Evaluating Text Segmentation},
  school    = {University of Ottawa},
  year      = {2013}
}
```
Bibliography


Python Module Index

S
segeval, 6
Index

A
actual_agreement_linear() (in module segeval), 15
artstein_poesio_bias_linear() (in module segeval), 16
Average (class in segeval), 18

B
boundary_confusion_matrix() (in module segeval), 13
boundary_edit_distance() (in module segeval), 13
boundary_similarity() (in module segeval), 12
boundary_statistics() (in module segeval), 11
boundary_string_from_masses() (in module segeval), 16
BoundaryFormat (class in segeval), 11

C
classes() (segeval.ConfusionMatrix method), 18
compute_window_size() (in module segeval), 14
ConfusionMatrix (class in segeval), 18
convert_masses_to_positions() (in module segeval), 16
convert_nltk_to_masses() (in module segeval), 16
convert_positions_to_masses() (in module segeval), 16
copy() (segeval.Dataset method), 16

D
Dataset (class in segeval), 16

F
Field (class in segeval), 17
fleiss_kappa_linear() (in module segeval), 15
fleiss_pi_linear() (in module segeval), 15
fmeasure() (in module segeval), 18

I
input_linear_mass_json() (in module segeval), 17
input_linear_mass_tsv() (in module segeval), 17

L
load_nested_folders_dict() (in module segeval), 17

O
output_linear_mass_json() (in module segeval), 17

P
pk() (in module segeval), 14
precision() (in module segeval), 17

R
recall() (in module segeval), 18

S
segeval (module), 6, 11
segmentation_similarity() (in module segeval), 12
SegmentationType (class in segeval), 17
summarize() (in module segeval), 18

W
weight_a() (in module segeval), 13
weight_s() (in module segeval), 13
weight_s_scale() (in module segeval), 14
weight_t() (in module segeval), 14
weight_t_scale() (in module segeval), 14
window_diff() (in module segeval), 15