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.

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

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

CHAPTER 1

Feature Support

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' π *[Fleiss1971]* (i.e., Siegel and Castellan's K *[SiegelCastellan1988]*)
- Fleiss' κ *[DaviesFleiss1982]*

CHAPTER 2

User Guide

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 ones segmentation data and methods easier. It implements the new segmentation comparison metrics detailed in *[Fournier2013]* and *[Fournier2013b]*.

License

TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

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.

```
{
        "items": {
                "stargazer": {
                        "1": [2,3,3,1,3,6,3],
                        "2": [2,8,2,4,2,3],
                        "3": [2,1,2,3,1,3,1,3,2,2,1],
                        "4": [2,1,4,1,1,3,1,4,3,1],
                        "5": [3,2,4,3,5,4],
                        "6": [2,3,4,2,2,5,3],
                        "7": [2,3,2,2,3,1,3,2,3]
                }
        },
        "segmentation_type": "linear"
}
```

Begin by importing the SegEval module:

```
>>> import segeval
```

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

```
>>> 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:

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

Segmentations can then be compared using functions such as:

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

Other metrics are also available, including:

```
>>> 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:

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

This produces:

```
{
    'matches': [1, 1, 1], # List of matching boundary types
    'boundaries_all': 11,
    'pbs': 20, # Potential boundaries
```

```
    'transpositions': [Transposition(start=8, end=9, type=1)],
    'full_misses': [1, 1, 1, 1, 1], # List of full miss boundary types
    'additions': [Addition(type=1, side='b'), Addition(type=1, side='a'),␣
↪Addition(type=1, side='a')],
    'count_edits': Decimal('3.5'), # Scaled count of all edits
    'substitutions': []
}
```

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:

```
>>> hypothesis = (2,6,4,2,4,3)
>>> reference = dataset['stargazer']['1']
>>> segeval.boundary_similarity(hypothesis, reference)
Decimal('0.571428571428571428571428571')
```

Some traditional segmentation comparison metrics can also be used:

```
>>> segeval.window_diff(hypothesis, reference)
Decimal('0.315789473684210526315789473 7')
>>> segeval.pk(hypothesis, reference)
Decimal('0.263157894736842105263157894 7')
```

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

```
>>> 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:

```
>>> segeval.precision(confusion_matrix)
Decimal('0.571428571428571428571428571')
>>> segeval.recall(confusion_matrix)
Decimal('0.571428571428571428571428571')
>>> segeval.fmeasure(confusion_matrix)
Decimal('0.727272727272727272727272727')
```

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:

```
>>> 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('0.583333333333333333333333333'),
    'stargazer,1,h1': Decimal('0.571428571428571428571428571'),
    'stargazer,1,h2': Decimal('0.388888888888888888888888889'),
    'stargazer,6,h2': Decimal('0.388888888888888888888888889'),
    'stargazer,7,h2': Decimal('0.318181818181818181818181818'),
    'stargazer,7,h1': Decimal('0.5'),
```

```
    'stargazer,5,h1': Decimal('0.41666666666666666666666666667'),
    'stargazer,5,h2': Decimal('0.375'),
    'stargazer,2,h1': Decimal('0.428571428571428571428571428571'),
    'stargazer,2,h2': Decimal('0.333333333333333333333333333333'),
    'stargazer,4,h2': Decimal('0.363636363636363636363636363636'),
    'stargazer,4,h1': Decimal('0.444444444444444444444444444444')
}
```

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:

```
>>> import segeval
>>> dataset = segeval.HEARST_1997_STARGAZER
>>> segeval.actual_agreement_linear(dataset)
Decimal('0.530054644808743169398907103 8')
```

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

```
>>> segeval.actual_agreement_linear(dataset, fnc_compare=segeval.segmentation_
↪similarity)
Decimal('0.795238095238095238095238095 2')
```

If instead we want a chance-corrected inter-coder agreement coefficient, Fleiss' κ and π adapted to use *`boundary_similarity()`* can be used:

```
>>> segeval.fleiss_kappa_linear(dataset)
Decimal('0.441491088906825498436731702 3')
>>> segeval.fleiss_pi_linear(dataset)
Decimal('0.440541243819932344522508456 9')
```

CHAPTER 3

API Documentation

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]*.

`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.

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]*.

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

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

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

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

`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, 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()*)

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]*.

`segeval.`**`segmentation_similarity`**(*segmentation_a*, *segmentation_b*, ***kwargs*)
For parameters see *boundary_similarity()*

`segeval.`**`segmentation_similarity`**(*dataset*, ***kwargs*)
For parameters see *boundary_similarity()*

segeval.**segmentation_similarity**()
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]*.

segeval.**boundary_edit_distance**(*boundary_string_a*, *boundary_string_b*, *n_t=2*)
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]*.

---

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

Parameters **segmentation_*** (*segmentation*) – Segmentation of a particular format; see *BoundaryFormat*

segeval.**boundary_confusion_matrix**(*dataset*, ***kwargs*)

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

segeval.**boundary_confusion_matrix**(**args*, ***kwargs*)

Weighting Functions

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

segeval.**weight_a**(*additions*)
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, demoninators, 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]*.

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

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

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.

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()*)

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' κ (or multi-κ), originally proposed in *[DaviesFleiss1982]*. For 2 coders, this is equivalent to Cohen's κ *[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 `[(),(),(),(),(1),(),(),(1),(),(),(),()]`.

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]` becomes `[5,3,5]`.

Parameters **segments** (*tuple*) – Ordered sequence of which segments a unit belongs to.

Deprecated 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]`.

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 manipuation.

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*, *dataset=None*, *prepend_item=[]*)
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.

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()*).

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.

class segeval.**ConfusionMatrix**
A dict()-like representation of a confusion matrix offering some automation. To access/store values, use: matrix[predicted][actual].

classes()
Retrieve the set of all classes.

CHAPTER 4

Support

If you have any suggestions, problems, or difficulties, please log an issue, or contact me.

CHAPTER 5

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:

```
@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}
}
```

CHAPTER 6

References

Bibliography

[ArtsteinPoesio2008] Ron Artstein and Massimo Poesio. 2008. **Inter-coder agreement for computational linguistics**. Computational Linguistics, 4(4):555-596. MIT Press.

[Baker1990] David Baker. 1990. **Stargazers look for life**. South Magazine 117, 76–77. South Publications.

[BeefermanBerger1999] Doug Beeferman and Adam Berger. 1999. **Statistical models for text segmentation**. Machine learning, 34(1–210. Springer Netherlands.

[Cohen1960] Jacob Cohen. 1960. **A Coefficient of Agreement for Nominal Scales**. Educational and Psychological Measurement, 20(1):37-46.

[Collins1868] Wilkie Collins. 1868. **The Moonstone**. Tinsley Brothers.

[DaviesFleiss1982] Mark Davies and Joseph L. Fleiss. 1982. **Measuring agreement for multinomial data**. Biometrics, 38(4):1047-1051.

[Fleiss1971] Joseph L. Fleiss. 1971. **Measuring nominal scale agreement among many raters**. Psychological Bulletin, 76(5):378-382.

[Fournier2013] Chris Fournier. 2013. **Evaluating Text Segmentation using Boundary Edit Distance**. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics. **To appear**.

[Fournier2013b] Chris Fournier. 2013. **Evaluating Text Segmentation**. Master's Thesis. University of Ottawa.

[FournierInkpen2012] Chris Fournier and Diana Inkpen. 2012. **Segmentation Similarity and Agreement**. Proceedings of Human Language Technologies: The 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics. (HLT '12). Association for Computational Linguistics.

[Hearst1997] Marti A. Hearst. 1997. **TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages**. Computational Linguistics, 23(1):33-64.

[KazantsevaSzpakowicz2012] Kazantseva, A. & Szpakowicz, S. (2012), **Topicalsegmentation: a study of human performance**. Proceedings of Human Language Technologies: The 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics. (HLT '12). Association for Computational Linguistics.

[LamprierEtAl2007] Sylvain Lamprier, Tassadit Amghar, Bernard Levrat, and Frederic Saubion 2007. **On evaluation methodologies for text segmentation algorithms**. Proceedings of the 19th IEEE International Conference on Tools with Arificial Intelligence, 2:19–26. IEEE Computer Society.

[PevznerHearst2002] Lev Pevzner and Marti A. Hearst. 2002. **A critique and improvement of an evaluation metric for text segmentation**. Computational Linguistics, 28(1):19–36. MIT Press, Cambridge, MA, USA.

[Scott1955] William A. Scott. 1955. **Reliability of content analysis: The case of nominal scale coding**. Public Opinion Quarterly, 19(3):321-325.

[SiegelCastellan1988] Sidney Siegel and N. John Castellan, Jr. 1988. **Non-parametric Statistics for the Behavioral Sciences**. 2nd Edition, Castellanhapter 9.8. McGraw-Hill.

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