evalmate Documentation

Release 0.3.0

buec

Nov 21, 2018

Notes

Evalmate is a set of tools for evaluate audio related machine learning tasks.

CHAPTER 1

Installation

Install the latest stable version:

```
pip install evalmate
```

Install the latest development version:

```
pip install git+https://github.com/ynop/evalmate.git
```

CHAPTER 2

Changelog

2.1 Next Version

Breaking changes

- Refactoring of all elements, so that it is more obvious which aligner is used for which evaluator and confusion.

New Features

- Introduced False Rejection Rate, False Alarm Rate, Term-Weight Value for the Keyword Spotting task.
- Evaluator for the Automatic Speech Recognition Task *evalmate.evaluator.ASREvaluator*.

2.2 v0.2.0

New Features

- Introduced *evalmate.evaluator.Outcome* to have a common input structure for reference and hypothesis.
- With *evalmate.evaluator.LabelSet* more statistics on reference and hypothesis can be computed. Label-Sets are created via *evalmate.evaluator.Outcome* class.

2.3 v0.1.0

Initial release

CHAPTER 3

evalmate.evaluator

This module implements the top-level functionality for performing the evaluation for the different tasks. For every task there is an Evaluator (extends *Evaluator*) and an Evaluation (extends *Evaluation*. The Evaluator is the is class responsible to perform the evaluation and the Evaluation is the output, which contains the aligned labels/segments and depending on the task further data like word confusions.

3.1 Base

class evalmate.evaluator.**Evaluation**(*ref_outcome*, *hyp_outcome*)

Base class for evaluation results.

Variables

- **ref_outcome** (Outcome) – The outcome of the ground-truth/reference.
- **hyp_outcome** (Outcome) – The outcome of the system-output/hypothesis.

get_report(*template=None*)

Generate and return a report.

Parameters template (*str*) – Name of the Jinja2 template to use. If None, the `default_template()` is used. All available templates are in the `report_templates` folder.

Returns The rendered report.

Return type str

template_data

Return a dictionary that contains objects/values to use in the rendering template.

write_report(*path*, *template=None*)

Write the report to the given path.

Parameters

- **path** (*str*) – Path to write the report to.

- **template** (*str*) – Name of the Jinja2 template to use. If None, the `default_template()` is used. All available templates are in the `report_templates` folder.

class `evalmate.evaluator.`**`Evaluator`**

Base class for a evaluator.

Provides methods for reading outcomes in different ways. The evaluator for a specific class then has to implement `do_evaluate`, which performs the evaluation on ref and hyp outcome.

classmethod default_label_list_idx()

Define the default label-lists which is used when reading a corpus.

do_evaluate(*ref*, *hyp*)

Create the evaluation result of the given hypothesis compared to the given reference (ground truth).

Parameters

- **ref** (`Outcome`) – The ground-truth/reference outcome.
- **hyp** (`Outcome`) – The system-output/hypothesis outcome.

Returns The evaluation results.

Return type *Evaluation*

evaluate(*ref*, *hyp*, *label_list_idx=None*)

Create the evaluation result of the given hypothesis compared to the given reference (ground truth). There are different possibilities of input:

- ref = Outcome / hyp = Outcome: Both ref and hyp are *Outcome* instances. See `do_evaluate`
- ref = Corpus / hyp = dict: The dict contains label-lists which are compared against the corpus. See `evaluate_label_lists_against_corpus`
- ref = LabelList / hyp = LabelList: Ref label-list is compared against the other. See `evaluate_label_lists`

Parameters

- **ref** (*LabelList, Corpus*) – A label-list, a corpus.
- **hyp** (*LabelList, dict*) – A label-list, a dict.
- **label_list_idx** (*str*) – The label-list to use when reading from a corpus.

Returns The evaluation results.

Return type *Evaluation*

evaluate_label_lists(*ll_ref*, *ll_hyp*, *duration=None*)

Create Evaluation for ref and hyp label-list. If the duration is not provided some metrics cannot be used.

Parameters

- **ref** (*LabelList*) – A label-list.
- **hyp** (*LabelList*) – A label-list.
- **duration** (*float*) – The duration of the utterance, that belongs to the label-lists.

Returns The evaluation results.

Return type *Evaluation*

evaluate_label_lists_against_corpus(*corpus*, *label_lists*, *label_list_idx=None*)
Create Evaluation for the given corpus.

Parameters

- **corpus** (`Corpus`) – A corpus containing the reference label-lists.
- **label_lists** (`Dict`) – A dictionary containing label-lists with the utterance-idx as key. The utterance-idx is used to find the corresponding reference label-list in the corpus.
- **label_list_idx** (`str`) – The idx of the label-lists to use as reference from the corpus. If None, *cls.default_label_list_idx* is used.

Returns The evaluation results.

Return type *Evaluation*

3.2 Outcome

class `evalmate.evaluator.`**Outcome**(*label_lists=None*, *utterance_durations=None*)
An outcome represents the annotation/labels/transcriptions of a dataset/corpus for a given task. This can be either the ground truth/reference or the system output/hypothesis.

If no durations are provided or duration for some utterances are missing, some methods may not work or throw exceptions.

Variables

- **label_lists** (`dict`) – Dictionary containing all label-lists with the utterance-idx/sample-idx as key.
- **utterance_durations** (`dict`) – Dictionary (utterance-idx/duration) containing the durations of all utterances.

all_values
Return a set of all values, occurring in the outcome.

label_set()
Return a label-set containing all labels.

label_set_for_value(*value*)
Return a label-set containing all labels, where the value is *value*.

Parameters **value** (`str`) – The value to filter.

Returns Label-set containing all labels with the given value.

Return type *LabelSet*

total_duration
Return the duration of all utterances together.

Notes

Only works if for all utterances, the durations are provided.

class `evalmate.evaluator.`**LabelSet**(*labels=None*)
Class to collect a bunch of labels. This is used to compute statistics over a defined set of labels.

For example we want to compute the average length of all labels with the value 'music'. We can then collect all these in a label-set and perform the computation.

count
Return the number of labels.

label_lengths
Return a list containing all label lengths.

length_max
Return the length of the longest label.

length_mean
Return the mean length of all labels.

length_median
Return the median of all label lengths.

length_min
Return the length of the shortest label.

length_variance
Return the variance of all label lengths.

3.3 Segment

class evalmate.evaluator.**SegmentEvaluation**(*ref_outcome*, *hyp_outcome*, *utt_to_segments*)
Result of an evaluation of a segment-based alignment.

Parameters **utt_to_segments** (*dict*) – Dict of lists with *evalmate.alignment.Segment*. Key is the utterance-idx.

Variables

- **ref_outcome** (Outcome) – The outcome of the ground-truth/reference.
- **hyp_outcome** (Outcome) – The outcome of the system-output/hypothesis.
- ***confusion*** (AggregatedConfusion) – Confusion result

segments
Return a list of all segment (from all utterances together).

template_data
Return a dictionary that contains objects/values to use in the rendering template.

class evalmate.evaluator.**SegmentEvaluator**(*aligner=None*)
Evaluation of an alignment based on segments.

Parameters **aligner** (SegmentAligner) – An instance of an event-aligner to use. If not given, the alignment.InvariantSegmentAligner is used.

classmethod default_label_list_idx()
Define the default label-lists which is used when reading a corpus.

do_evaluate(*ref*, *hyp*)
Create the evaluation result of the given hypothesis compared to the given reference (ground truth).

Parameters

- **ref** (Outcome) – The ground-truth/reference outcome.
- **hyp** (Outcome) – The system-output/hypothesis outcome.

Returns The evaluation results.

Return type *Evaluation*

static flatten_overlapping_labels(*aligned_segments*)
Check all segments for overlapping labels. Overlapping means there are multiple reference or multiple hypothesis labels in a segment.

Parameters **aligned_segments** (`List`) – List of segments.

Returns List of segments where ref and hyp is a single label.

Return type list

Raises `ValueError` – A segment contains overlapping labels.

3.4 Event

class evalmate.evaluator.**EventEvaluation**(*ref_outcome*, *hyp_outcome*, *utt_to_label_pairs*)
Result of an evaluation of any event-based alignment.

Parameters **utt_to_label_pairs** (*dict*) – Key is the utterance-id, value is a list of *evalmate.alignment.LabelPair*.

Variables

- **ref_outcome** (`Outcome`) – The outcome of the ground-truth/reference.
- **hyp_outcome** (`Outcome`) – The outcome of the system-output/hypothesis.
- ***confusion*** (`AggregatedConfusion`) – Confusion statistics

label_pairs
Return a list of all label-pairs (from all utterances together).

template_data
Return a dictionary that contains objects/values to use in the rendering template.

class evalmate.evaluator.**EventEvaluator**(*aligner*)
Class to compute evaluation results for any event-based alignment.

Parameters **aligner** (`EventAligner`) – An instance of an event-aligner to use.

classmethod default_label_list_idx()
Define the default label-lists which is used when reading a corpus.

do_evaluate(*ref*, *hyp*)
Create the evaluation result of the given hypothesis compared to the given reference (ground truth).

Parameters

- **ref** (`Outcome`) – The ground-truth/reference outcome.
- **hyp** (`Outcome`) – The system-output/hypothesis outcome.

Returns The evaluation results.

Return type *Evaluation*

3.5 KWS

class evalmate.evaluator.**KWSEvaluation**(*ref_outcome*, *hyp_outcome*, *utt_to_label_pairs*)
Result of an evaluation of a keyword spotting task.

Parameters **utt_to_label_pairs** (*dict*) – Key is the utterance-id, value is a list of *evalmate.alignment.LabelPair*.

Variables

- **ref_outcome** (Outcome) – The outcome of the ground-truth/reference.
- **hyp_outcome** (Outcome) – The outcome of the system-output/hypothesis.
- ***confusion*** (AggregatedConfusion) – Confusion statistics

false_alarm_rate (*keyword=None*)

The False Alarm Rate (FAR) is the percentage of detections, where no keyword is according to the ground truth. If no keyword is given the mean FAR is calculated over all keywords. This rate is relative to the duration of all utterances.

To calculate this, we need to know the number of times a keyword could be wrongly inserted. We assume that every keyword takes one second to approximate this value.

Parameters **keyword** (*str*) – If not None, only the FFR for this keyword is returned.

Returns A rate between 0 and 1

Return type float

false_rejection_rate (*keyword=None*)

The False Rejection Rate (FRR) is the percentage of misses of all occurrences in the ground truth. If no keyword is given the mean FRR is calculated over all keywords.

Parameters **keyword** (*str*) – If not None, only the FFR for this keyword is returned.

Returns A rate between 0 and 1

Return type float

keywords ()

Return a list of all keywords occurring in the reference outcome.

term_weighted_value (*keyword=None*)

Computes the Term-Weighted Value (TWV).

Note: The TWV is implemented according to OpenKWS 2016 Evaluation Plan

Parameters **keyword** (*str*) – If None, computes the TWV over all keywords, otherwise only for the given keyword.

Returns The TWV in the range 1 to -inf

Return type float

class evalmate.evaluator.**KWSEvaluator** (*aligner=None*)

Class to retrieve evaluation results for a keyword spotting task.

Parameters **aligner** (EventAligner) – An instance of an event-aligner to use. If not given the *evalmate.alignment.BipartiteMatchingAligner* is user.

classmethod default_label_list_idx ()

Define the default label-lists which is used when reading a corpus.

do_evaluate (*ref*, *hyp*)

Create the evaluation result of the given hypothesis compared to the given reference (ground truth).

Parameters

- **ref** (Outcome) – The ground-truth/reference outcome.
- **hyp** (Outcome) – The system-output/hypothesis outcome.

Returns The evaluation results.

Return type *Evaluation*

3.6 ASR

class evalmate.evaluator.**ASREvaluation**(*ref_outcome*, *hyp_outcome*, *utt_to_label_pairs*)
Result of an evaluation of a automatic speech recognition task.

Parameters **utt_to_label_pairs** (*dict*) – Key is the utterance-id, value is a list of *evalmate.alignment.LabelPair*.

Variables

- **ref_outcome** (Outcome) – The outcome of the ground-truth/reference.
- **hyp_outcome** (Outcome) – The outcome of the system-output/hypothesis.
- ***confusion*** (AggregatedConfusion) – Confusion statistics

class evalmate.evaluator.**ASREvaluator**(*aligner=None*)
Class to retrieve evaluation results for a automatic speech recognition task.

Parameters **aligner** (EventAligner) – An instance of an event-aligner to use. If not given, the alignment.LevenshteinAligner is used.

classmethod default_label_list_idx()
Define the default label-lists which is used when reading a corpus.

do_evaluate(*ref*, *hyp*)
Create the evaluation result of the given hypothesis compared to the given reference (ground truth).

Parameters

- **ref** (Outcome) – The ground-truth/reference outcome.
- **hyp** (Outcome) – The system-output/hypothesis outcome.

Returns The evaluation results.

Return type *Evaluation*

static tokenize(*ll*, *overlap_threshold=0.1*)
Tokenize a label-list and return a new label-list with a separate label for every token.

CHAPTER 4

evalmate.confusion

This module contains classes for computing confusion statistics.

4.1 Confusion

class evalmate.confusion.**Confusion**

Base class that provides methods for computing common metrics.

accuracy

Accuracy = correct / (total + insertions)

correct

Amount that is correct.

Example

```
>>> ref = 'xxx'
>>> hyp = 'xxx'
```

deletions

Amount that is deleted.

Example

```
>>> ref = 'xxx'
>>> hyp = None
```

error_rate

ErrorRate = (substitutions + deletions + insertions) / total

`f_measure`(*beta=1*)
F-Measure see https://en.wikipedia.org/wiki/Precision_and_recall

`false_negatives`
Amount of false negatives (No indication of precence, when it should be present).

Note: Equal to 'self.total - self.correct'

`false_positives`
Amount of false positives (Indications of presence, when it is not present).

Note: Equal to *self.insertions + self.substitutions_out*

`insertions`
Amount that is inserted.

Example

```
>>> ref = None
>>> hyp = 'xxx'
```

`precision`
Precision = tp / (fp + tp)

`recall`
Recall = tp / (fn + tp)

`substitutions`
Amount that is substituted.

If this stats are representing stats for a specific instance (e.g. occurrence of the word 'hello') `substitutions` is the amount where the specific instance was substituted with some other instance/event. If not it is not necessary to designate which event/instance substitutes which event/instance.

Example

```
>>> ref = 'xxx'
>>> hyp = 'yyy'
```

`substitutions_out`
Amount that is substituted.

If this stats are representing stats for a specific instance (e.g. occurrence of the word 'hello') `substitutions_out` is the amount where the specific instance was output, when some other event/instance was expected (reference). If not it is equal to `substitutions`.

Example

```
>>> ref = 'yyy'
>>> hyp = 'xxx'
```

total
Return the total amount based on the reference system.

Note: Equal to 'self.correct + self.deletions + self.substitutions'

true_positives
Amount of true positives (Correct indications).

Note: Equal to *self.correct*

4.2 SegmentConfusion

class evalmate.confusion.**SegmentConfusion**(*value*)
Class to represent confusions of a specific instance (e.g. some class) based on segments. The insertions, deletions and so on represent the time in seconds the instance was confused (or not).

Argument: value (str): The value of the instance (e.g. the class "speech")

Variables

- **correct_segments** (*list*) – (List of Segment) Segments that are correct (ref == hyp).
- **insertion_segments** (*list*) – (List of Segment) Segments that are insertions (ref = None, hyp = 'value').
- **deletion_segments** (*list*) – (List of Segment) Segments that are deletions (ref = 'value', hyp = None)
- **substitution_segments** (*Dict*) – Segments that are substitutions with other values (ref = 'value', hyp = 'other-value'). Dict holding a list for every *other-value*.
- **substitution_out_segments** (*Dict*) – Segments that are substitutions of other values (ref = 'other-value', hyp = 'value'). Dict holding a list for every *other-value*.

correct
Amount that is correct.

Example

```
>>> ref = 'xxx'
>>> hyp = 'xxx'
```

deletions
Amount that is deleted.

Example

```
>>> ref = 'xxx'
>>> hyp = None
```

insertions
Amount that is inserted.

Example

```
>>> ref = None
>>> hyp = 'xxx'
```

substitutions
Amount that is substituted.

If this stats are representing stats for a specific instance (e.g. occurrence of the word 'hello') `substitutions` is the amount where the specific instance was substituted with some other instance/event. If not it is not necessary to designate which event/instance substitutes which event/instance.

Example

```
>>> ref = 'xxx'
>>> hyp = 'yyy'
```

substitutions_out
Amount that is substituted.

If this stats are representing stats for a specific instance (e.g. occurrence of the word 'hello') `substitutions_out` is the amount where the specific instance was output, when some other event/instance was expected (reference). If not it is equal to `substitutions`.

Example

```
>>> ref = 'yyy'
>>> hyp = 'xxx'
```

4.3 EventConfusion

class evalmate.confusion.**EventConfusion**(*value*)
Class to represent confusions of a specific instance (e.g. some class) based on label-to-label alignment. The insertions, deletions and so on represent the number of times a label was confused (or not).

Argument: value (str): The value of the instance (e.g. the class "speech")

Variables

- **correct_pairs** (*list*) – (List of LabelPair) Correct matches.
- **insertion_pairs** (*list*) – (List of LabelPair) Insertions (ref = None, hyp = value)
- **deletion_pairs** (*list*) – (List of LabelPair) Deletions (ref = value, hyp = None)
- **substitution_pairs** (*Dict*) – Substitutions with other values (ref = value, hyp = other-value). Dict holding a list for every *other-value*.
- **substitution_out_pairs** (*Dict*) – Substitutions from other values (ref = other-value, hyp = value) Dict holding a list for every *other-value*.

correct
Amount that is correct.

Example

```
>>> ref = 'xxx'
>>> hyp = 'xxx'
```

deletions
Amount that is deleted.

Example

```
>>> ref = 'xxx'
>>> hyp = None
```

insertions
Amount that is inserted.

Example

```
>>> ref = None
>>> hyp = 'xxx'
```

substitutions
Amount that is substituted.

If this stats are representing stats for a specific instance (e.g. occurrence of the word 'hello') `substitutions` is the amount where the specific instance was substituted with some other instance/event. If not it is not necessary to designate which event/instance substitutes which event/instance.

Example

```
>>> ref = 'xxx'
>>> hyp = 'yyy'
```

substitutions_by_count()
Return a list of tuples (Substituted-value, Number-of-substitutions) ordered by number of substitutions descending.

Returns List of tuples.

Return type list

substitutions_out
Amount that is substituted.

If this stats are representing stats for a specific instance (e.g. occurrence of the word 'hello') `substitutions_out` is the amount where the specific instance was output, when some other event/instance was expected (reference). If not it is equal to `substitutions`.

Example

```
>>> ref = 'yyy'
>>> hyp = 'xxx'
```

4.4 AggregatedConfusion

class evalmate.confusion.**AggregatedConfusion**

Class to aggregate multiple confusions.

Variables **instances** (*dict*) – Dictionary containing the aggregated confusions.

correct

Amount that is correct.

Example

```
>>> ref = 'xxx'
>>> hyp = 'xxx'
```

deletions

Amount that is deleted.

Example

```
>>> ref = 'xxx'
>>> hyp = None
```

insertions

Amount that is inserted.

Example

```
>>> ref = None
>>> hyp = 'xxx'
```

substitutions

Amount that is substituted.

If this stats are representing stats for a specific instance (e.g. occurrence of the word 'hello') `substitutions` is the amount where the specific instance was substituted with some other instance/event. If not it is not necessary to designate which event/instance substitutes which event/instance.

Example

```
>>> ref = 'xxx'
>>> hyp = 'yyy'
```

substitutions_out

Amount that is substituted.

If this stats are representing stats for a specific instance (e.g. occurrence of the word 'hello') `substitutions_out` is the amount where the specific instance was output, when some other event/instance was expected (reference). If not it is equal to `substitutions`.

Example

```
>>> ref = 'yyy'
>>> hyp = 'xxx'
```

CHAPTER 5

evalmate.alignment

This module contains functionality for aligning labels of a ground truth with the labels of a system output.

5.1 Base classes

All aligners are based either on *EventAligner* or *SegmentAligner*. The base classes are mainly distinguished by the type of the alignment they return. While the *EventAligner* returns a mapping between complete labels, the *SegmentAligner* returns segments, that can span over parts of labels.

class evalmate.alignment.**EventAligner**

Abstract class for aligner classes that return a mapping between labels (events).

An alignment is a mapping between labels from the ground truth (ref) and the system output (hyp). If there is no matching label in the system output for a label in the ground truth, it has to be aligned to `None` and vice versa. A single label can be aligned to multiple other labels.

align (*ref*, *hyp*)

Return an alignment between the labels of the two label-lists.

Parameters

- **ref** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels of the ground truth.
- **hyp** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels of the system output.

Returns A list of *evalmate.alignment.LabelPair*. Every pair contains one label from the ground truth and one from the system output, that are aligned. One of them also can be `None`.

Return type list

class evalmate.alignment.**SegmentAligner**

Abstract class for aligner classes that align labels in segments.

An alignment is represented as a list of Segments with start/end-time and the labels from the ground truth and the system output, that are within this segment.

align (*ref*, *hyp*)
Return an alignment of segments.

Parameters

- **ref** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels of the ground truth.
- **hyp** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels of the system output.

Returns A list of `evalmate.utils.structure.Segment`. Every segment has start/end-time and two lists of labels that are contained in the segment (one for the ground truth and one for the system output).

Return type list

5.2 Time-Based

Align labels based on some distance metric based on their start/endtimes.

class evalmate.alignment.**BipartiteMatchingAligner** (*start_delta_threshold=0.5*, *end_delta_threshold=-1*, *non_overlap_penalty_weight=1*, *substitution_penalty=2*, *insertion_penalty=10*, *deletion_penalty=10*)
Create event-based alignment, based on bipartite matching.

1. In a first step for every possible label-pair between ref and hyp, it is decided if a mapping of such a pair is possible. This decision is based on the `start_delta_threshold` and `end_delta_threshold`.
2. Using penalty and weight parameters, for every pair a penalty is computed for aligning the pair.
3. From all the pairs and the computed probabilities, the best alignment is computed using bipartite matching. So that every label only occurs once in the final alignment.

Parameters

- **start_delta_threshold** (*float*) – Temporal tolerance of the start time in seconds. If the delta between the starts of the two labels is greater it is not a matching pair.
- **end_delta_threshold** (*float*) – Temporal tolerance of the end time in seconds. If the delta between the ends of the two labels is greater it is not a matching pair. If < 0 the end time is not checked at all.
- **non_overlap_penalty_weight** (*float*) – Weight-factor of penalty for the non-overlapping ratio between two labels.
- **substitution_penalty** (*float*) – Penalty for aligning two labels with different values.
- **deletion_penalty** (*float*) – Penalty for aligning a reference-label with no hypothesis-label.
- **insertion_penalty** (*float*) – Penalty for aligning a hypothesis-label with no reference-label.

align (*ll_ref*, *ll_hyp*)
Return an alignment between the events of the given label-lists.

Parameters

- **ref** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels (events) of the ground truth.
- **hyp** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels (events) of the system output.

Returns A list of *evalmate.alignment.LabelPair*. Every pair contains one label (event) from the ground truth and one from the system output, that are aligned. One of them also can be `None`.

Return type list

class evalmate.alignment.**FullMatchingAligner** (*min_overlap=0*)
Event-based alignment, where all possible matches are returned. So a single label can occur multiple times, but with a different counterpart.

Parameters **min_overlap** (*float*) – Number of seconds the segment of overlap has to be, to align two labels. If `0`, any overlap is accepted.

align (*ref*, *hyp*)
Return an alignment between the events of the given label-lists.

Parameters

- **ref** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels (events) of the ground truth.
- **hyp** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels (events) of the system output.

Returns A list of *evalmate.alignment.LabelPair*. Every pair contains one label (event) from the ground truth and one from the system output, that are aligned. One of them also can be `None`.

Return type list

5.3 Sequence-Based

Align labels only considering the ordering of the sequence.

class evalmate.alignment.**LevenshteinAligner** (*deletion_cost=3*, *insertion_cost=3*, *substitution_cost=4*, *custom_substitution_cost_function=None*)
Alignment of labels of two label-lists based on the Levenshtein distance (https://en.wikipedia.org/wiki/Levenshtein_distance).

This only takes the order of the labels into account, not the start and end-times.

Parameters

- **deletion_cost** (*float*) – Cost for a deletion in the alignment.
- **insertion_cost** (*float*) – Cost for a insertion in the alignment.
- **substitution_cost** (*float*) – Cost for a substitution in the alignment.

- **custom_substitution_cost_function** (*func*) – Function to calculate substitution cost depending on the elements. The function has to take two paramters (ref-label, hyp-label).

align (*reference*, *hypothesis*)

Return an alignment between the labels of the given label-lists.

Parameters

- **reference** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels of the ground truth.
- **hypothesis** (*audiomate.corpus.assets.LabelList*) – The label-list containing labels of the system output.

Returns A list of *evalmate.alignment.LabelPair*. Every pair contains one label from the ground truth and one from the system output, that are aligned. One of them also can be `None`.

Return type list

Example

```
>>> from audiomate.corpus import assets
>>>
>>> reference = assets.LabelList(labels=[
>>>     assets.Label('a'),
>>>     assets.Label('b'),
>>>     assets.Label('c')
>>> ])
>>> hypothesis = assets.LabelList(labels=([
>>>     assets.Label('a'),
>>>     assets.Label('c')
>>> ])
>>>
>>> LevenshteinAligner().align(reference, hypothesis)
[
    LabelPair(Label('a'), Label('a')),
    LabelPair(Label('b'), None),
    LabelPair(Label('c'), Label('c'))
]
```

5.4 Segment-Based

Align labels based on segments defined by start/end-time.

class evalmate.alignment.**InvariantSegmentAligner**

Create a segment-based alignment so that within every segment the same labels are active. So for example as reference we have a label-list as following.

```
>>> [    A    ]      [      B      ]     [       A      ]
>>>                                    [       E      ]
```

The output of some system (hypothesis) maybe as follows:

```
>>> [    Ax   ]       [   Ex  ]                              [   Ax  ]
```

Now the segments returned are created, so every segment represents some time range where the labels are equal.

```
>>>          S1       S2    S3     S4     S5         S6       S7    S8
>>>
>>> HYP  |    A    |       |  B  |  B   |     |       A       |    |     |
>>> HYP  |         |       |     |      |     |       E       |    |     |
>>> REF  |    Ax   |       |  Ex |      |     |               |    |  Ax |
```

align (*ll_ref, ll_hyp*)
Create segment based alignment.

Parameters

- **ll_ref** (*audiomate.corpus.assets.LabelList*) – The label-list with reference labels.
- **ll_hyp** (*audiomate.corpus.assets.LabelList*) – The label-list with hypothesis labels.

Returns A list of Segments.

Return type list

Example

```
>>> from audiomate.corpus import assets
>>>
>>> ref = assets.LabelList(labels=[
>>>     assets.Label('a', 0, 3),
>>>     assets.Label('b', 3, 6),
>>>     assets.Label('c', 7, 10)
>>> ])
>>>
>>> hyp = assets.LabelList(labels=[
>>>     assets.Label('a', 0, 3),
>>>     assets.Label('b', 4, 8),
>>>     assets.Label('c', 8, 10)
>>> ])
>>>
>>> InvariantSegmentAligner().align(ref, hyp)
[
    0 - 3 REF: [Label(a, 0, 3)] HYP: [Label(a, 0, 3)]
    3 - 4 REF: [Label(b, 3, 6)] HYP: []
    4 - 6 REF: [Label(b, 3, 6)] HYP: [Label(b, 4, 8)]
    6 - 7 REF: [] HYP: [Label(b, 4, 8)]
    7 - 8 REF: [Label(c, 7, 10)] HYP: [Label(b, 4, 8)]
    8 - 10 REF: [Label(c, 7, 10)] HYP: [Label(c, 8, 10)]
]
```

static create_event_list (*ll_ref, ll_hyp, time_threshold=0.01*)
Create an event list of all labels.

Parameters

- **ll_ref** (*LabelList*) – Reference labels.
- **ll_hyp** (*LabelList*) – Hypothesis labels.

- **time_threshold** (*float*) – If two event times are closer than this threshold the time of the earlier event is used for both events.

Returns List of list of tuples. Every tuple contains a time, type (start or end), ll_index (ref/hyp) and the label which is responsible for the event. It is sorted ascending by time.

Return type list

static set_absolute_end_of_labels(*label_list*)
If there are any labels where the end is defined as -1 (end of utterance), set the concrete time.

Parameters **label_list** (*LabelList*) – The label-list to process.

5.5 Utils

class evalmate.alignment.**Segment**(*start*, *end*, *ref=None*, *hyp=None*)
A class representing a segment within an alignment.

Parameters

- **start** (*float*) – The start time in seconds.
- **end** (*float*) – The end time in seconds.

Variables

- **ref** (*Label, list*) – List of or single reference label in the segment.
- **hyp** (*Label, list*) – List of or single hypothesis label in the segment.

class evalmate.alignment.**LabelPair**(*ref*, *hyp*)
Class to hold a pair of labels.

Variables

- **ref** (*Label*) – Reference label.
- **hyp** (*Label*) – Hypothesis label.

max_length()
Return the length of the longer value from ref and hyp.

padded_hyp_value()
Return the hypothesis value as string padded to the longer value of ref and hyp.

padded_ref_value()
Return the reference value as string padded to the longer value of ref and hyp.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index