ward-metrics Documentation

Release

Peter Hevesi

Feb 24, 2018

Contents:

CHAPTER 1

Introduction

Goal of this package is to provide an easy to use API for event-based evaluations in activity recognition problems.

As pointed out by *[WLG11]*, standard precision and recall values aren't sufficient to describe the nature of the errors in a dataset. Therefore the authors introduced new categories, metrics and visualisations for event-based evaluations.

The package implements the methods proposed by *[WLG11]* beside the typical precision and recall calculations. We provide also useful functions to interface with other modules of your activity recognition workflow and visualisation tools in the package.

1.1 Installation

Easiest way is to install it with pip (command line):

```
pip install ward-metrics
```

To update to the latest version you can call (command line):

```
pip install ward-metrics --upgrade
```

To import the package in the project your can simply write:

```
import wardmetrics
```

The package is currently only tested with python 3 (>=3.3).

For adapting the package to your needs, checkout the our Github repository.

1.2 References

CHAPTER 2

Descriptions

This is a short overview of the used naming conventions. For a more detailed discussion check out for example the paper *[WLG11]*.

Ground truth events: actual events that were annotated and will be considered as true instances of an activity. (However depending on the annotation quality they are not always 100% correct.)

Detected events: predicted events - typically results of a classifier's prediction step

2.1 Standard scores

Can be defined on a frame-by-frame basis. To calculate score values each frame is categorized with one of the following score labels:

True positives (TP): ground truth label for the current frame or segment equals the detected (predicted) label

False positives (FP): current frame or segment has a detected label for a particular class however the ground truth indicates that this class is not active

False negatives (FN): detected label is not equal to the ground truth when ground truth shows presence of the class

True negatives (TN): when neither the detected nor the corresponding ground truth label represents the class

Segments can be defined as a group of sequential frames where the assigned score labels don't change.

Precision:

$$precision = \frac{TP}{TP + FP}$$

Recall:

$$recall = \frac{TP}{TP + FN}$$

2.1.1 Calculation of event-based precision and recall

We can Assumption for True positive events False positive False negatives

TP_det, FP_det todo: add formula

As mentioned in *[WLG11]*, there are several issues with these methods for event-based evaluation, especially that the nature of the errors remain mostly hidden. The authors propose additional segment categories to better describe the results.

2.2 Detailed scores

2.2.1 Detailed scores for ground truth events

C - Correct: todo

F - fragmented: todo

M - merged: todo

FM - fragmented and merged: todo

D - deletion: todo practically equivalent of a false negative ground truth event

2.2.2 Detailed scores for detection events

C - Correct F' - fragmenting: ... M' - merging: ... FM' - fragmenting and merging: ... I' - insertion: ... basically equivalent of a false positive detection event

2.2.3 Detailed scores for segments

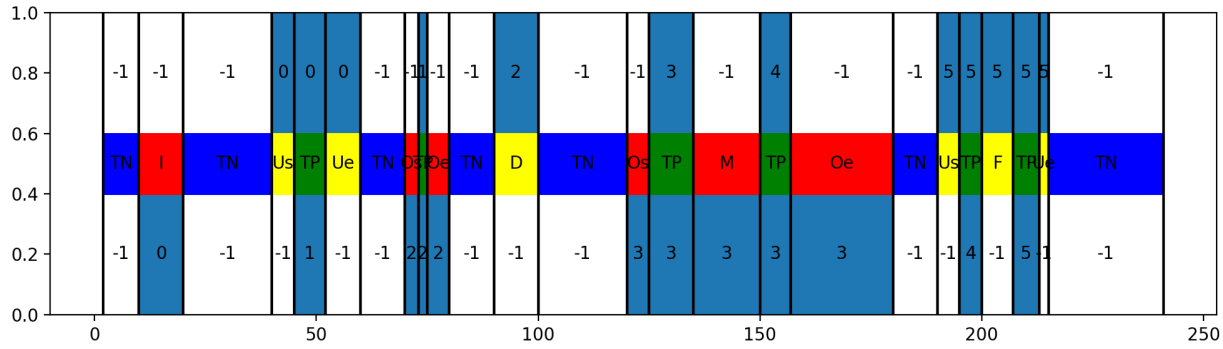
Detailed segment categories:

By using the detailed score categories, we can also evaluate the dataset on a frame-by-frame base receiving additional information on the event based errors.

To each segment (group of sequential frames with the same score label) we can assign one of the following categories:

- True positive - TP
- True negative - TN
- Insertion - I
- Merge - M,
- Deletion - D
- Fragmenting - F
- Start overfill - Os
- End overfill - Oe
- Start underfill - Us
- End underfill - Ue

As shown on the next figure:



Segment based metrics (2SET-metrics):

todo: add example plot

CHAPTER 3

Examples

todo: describe purpose of this section, use the following examples to get familiar with the API

3.1 Example 1: Segment detection and scoring

```
from wardmetrics.core_methods import eval_segments
from wardmetrics.visualisations import *
from wardmetrics.utils import *

ground_truth_test = [
    (40, 60),
    (70, 75),
    (90, 100),
    (125, 135),
    (150, 157),
    (187, 220),
]

detection_test = [
    (10, 20),
    (45, 52),
    (65, 80),
    (120, 180),
    (195, 200),
    (207, 213),
]

eval_start = 2
eval_end = 241

# Calculate segment results:
twoset_results, segments_with_scores, segment_counts, normed_segment_counts = eval_
↪segments(ground_truth_test, detection_test, eval_start, eval_end)
```

```
# Print results:
print_detailed_segment_results(segment_counts)
print_detailed_segment_results(normed_segment_counts)
print_twoset_segment_metrics(twoset_results)

# Access segment results in other formats:
print("\nAbsolute values:")
print("-----------------")
print(detailed_segment_results_to_list(segment_counts)) # segment scores as basic␣
↪python list
print(detailed_segment_results_to_string(segment_counts)) # segment scores as string␣
↪line
print(detailed_segment_results_to_string(segment_counts, separator=";", prefix="(",␣
↪suffix=")\n")) # segment scores as string line

print("Normed values:")
print("--------------")
print(detailed_segment_results_to_list(normed_segment_counts)) # segment scores as␣
↪basic python list
print(detailed_segment_results_to_string(normed_segment_counts)) # segment scores as␣
↪string line
print(detailed_segment_results_to_string(normed_segment_counts, separator=";", prefix=
↪"(", suffix=")\n")) # segment scores as string line

# Access segment metrics in other formats:
print("2SET metrics:")
print("-------------")
print(twoset_segment_metrics_to_list(twoset_results)) # twoset_results as basic␣
↪python list
print(twoset_segment_metrics_to_string(twoset_results)) # twoset_results as string␣
↪line
print(twoset_segment_metrics_to_string(twoset_results, separator=";", prefix="(",␣
↪suffix=")\n")) # twoset_results as string line

# Visualisations:
plot_events_with_segment_scores(segments_with_scores, ground_truth_test, detection_
↪test)
plot_segment_counts(segment_counts)
plot_twoset_metrics(twoset_results)
```

3.2 Example 2: Event-based evaluation

```
import wardmetrics
from wardmetrics.core_methods import eval_events
from wardmetrics.utils import *
from wardmetrics.visualisations import *

ground_truth_test = [
    (40, 60),
    (73, 75),
    (90, 100),
    (125, 135),
    (150, 157),
    (190, 215),
    (220, 230),
```

```
    (235, 250),
]

detection_test = [
    (10, 20),
    (45, 52),
    (70, 80),
    (120, 180),
    (195, 200),
    (207, 213),
    (221, 237),
    (239, 243),
    (245, 250),
]

print("Using version",  wardmetrics.__version__)

# Run event-based evaluation:
gt_event_scores, det_event_scores, detailed_scores, standard_scores = eval_
↪events(ground_truth_test, detection_test)

# Print results:
print_standard_event_metrics(standard_scores)
print_detailed_event_metrics(detailed_scores)

# Access results in other formats:
print(standard_event_metrics_to_list(standard_scores)) # standard scores as basic␣
↪python list, order: p, r, p_w, r_w
print(standard_event_metrics_to_string(standard_scores)) # standard scores as string␣
↪line, order: p, r, p_w, r_w)
print(standard_event_metrics_to_string(standard_scores, separator=";", prefix="(",␣
↪suffix=")\n")) # standard scores as string line, order: p, r, p_w, r_w

print(detailed_event_metrics_to_list(detailed_scores)) # detailed scores as basic␣
↪python list
print(detailed_event_metrics_to_string(detailed_scores)) # detailed scores as string␣
↪line
print(detailed_event_metrics_to_string(detailed_scores, separator=";", prefix="(",␣
↪suffix=")\n")) # standard scores as string line


# Show results:
plot_events_with_event_scores(gt_event_scores, det_event_scores, ground_truth_test,␣
↪detection_test, show=False)
plot_event_analysis_diagram(detailed_scores)
```

3.3 Example 3: Customizing plots

```
import wardmetrics
from wardmetrics.core_methods import eval_events
from wardmetrics.visualisations import *
from wardmetrics.utils import *

ground_truth_test = [
    (40, 60),
```

```
    (73, 75),
    (90, 100),
    (125, 135),
    (150, 157),
    (190, 215),
    (220, 230),
    (235, 250),
    (275, 292),
    (340, 368),
    (389, 410),
    (455, 468),
    (487, 512),
    (532, 546),
    (550, 568),
    (583, 612),
    (632, 645),
    (655, 690),
    (710, 754),
    (763, 785),
    (791, 812),
]

detection_test = [
    (10, 20),
    (45, 52),
    (70, 80),
    (120, 180),
    (195, 200),
    (207, 213),
    (221, 237),
    (239, 243),
    (245, 250),
]

print("Using version",  wardmetrics.__version__)

gt_event_scores, det_event_scores, detailed_scores, standard_scores = eval_
↪events(ground_truth_test, detection_test)

print_standard_event_metrics(standard_scores)
print(standard_event_metrics_to_list(standard_scores))
print(standard_event_metrics_to_string(standard_scores))
exit()

plot_events_with_event_scores(gt_event_scores, det_event_scores, ground_truth_test,␣
↪detection_test, show=False)

plot_event_analysis_diagram(detailed_scores, fontsize=8, use_percentage=True)
#plot_event_analysis_diagram(results)
```

CHAPTER 4

Core-methods

4.1 Segmentation and segment-based metrics

Similar (or equal) to frame by frame evaluations but using the detailed error scores additionally.

wardmetrics.core_methods.**eval_segments**(*ground_truth_events*, *detected_events*, *evaluation_start=None*, *evaluation_end=None*)

Segment-based evaluation (frame - length based)

Computes and scores segments and returns the occurrences of each error type in the overall dataset segments

Parameters

- **ground_truth_events** (*list of tuples (start, end) or lists [start, end]*) – numeric values (e.g. frame number or posix timestamp) for ground truth events' start and end times
- **detected_events** (*list of tuples (start, end) or lists [start, end]*) – numeric values (e.g. frame number or posix timestamp) for detected events' start and end times
- **evaluation_start** (*numeric value or None*) – This should be the first segment's start value. None indicates that start of the first event should be used.
- **evaluation_end** (*numeric value or None*) – This should be the first segment's start value. None indicates that start of the first event should be used.

Returns

- **twoset_results** (*dictionary*) – result for the 2SET metrics as a dictonary
- **segments_with_detailed_categories** (*list of tuples*) – list of detected segments including standard and detailed score categories
- **segment_counts** (*dictionary*) – frame counts/length of segments for each category
- **normed_segment_counts** (*dictionary*) – same as before but normed

4.2 Event-based metrics

wardmetrics.core_methods.**eval_events**(*ground_truth_events*, *detected_events*, *evaluation_start=None*, *evaluation_end=None*)

Event-based evaluation

Assigns scores to each ground truth and detection event and calculates statistics

Parameters

- **ground_truth_events** (*list of tuples (start, end) or lists [start, end]*) – numeric values (e.g. frame number or posix timestamp) for ground truth events' start and end times
- **detected_events** (*list of tuples (start, end) or lists [start, end]*) – numeric values (e.g. frame number or posix timestamp) for detected events' start and end times
- **evaluation_start** (*numeric value or None*) – This should be the first segment's start value. None indicates that start of the first event should be used.
- **evaluation_end** (*numeric value or None*) – This should be the first segment's start value. None indicates that start of the first event should be used.

Returns

- **gt_scores** (*list*) – score label for each ground truth event
- **detection_scores** (*list*) – score label for each detected event
- **detailed_score_statistics** (*dictionary*) – containing total number of events for each score category
- **standard_score_statistics** (*dictionary*) – precision and recall values (normal and weighted with event length) based on standard event scores (TP, FP, TN, FN)

CHAPTER 5

Visualisations

5.1 Event analysis diagram

wardmetrics.visualisations.**plot_event_analysis_diagram**(*event_results*, ***kwargs*)

Plot the event analysis diagram (EAD) for the given results

Visualisation of the distribution of specific error types either with the actual event count or showing the percentage of the total events. Elements of the plot can be adjusted (like color, fontsize etc.)

Parameters **event_results** (*dictionary*) – Dictionary containing event counts for "total_gt", "total_det", "D", "F", "FM", "M", "C", "M'", "FM'", "F'", "I'" as returned by core_methods.event_metrics' third value

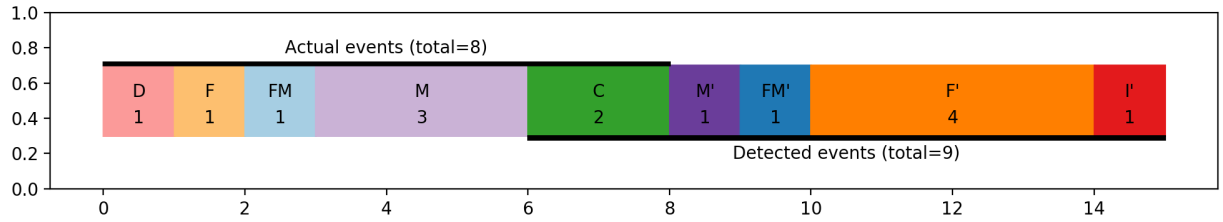
Keyword Arguments

- **fontsize** (*int*) – Size of the text inside the bar plot (Reduce the value if some event types are too short)
- **use_percentage** (*bool*) – whether percentage values or to show actual event counts on the chart (default: False)
- **show** (*bool*) – whether to call plt.show (blocking) or plt.draw() for later displaying (default: True)
- **color_deletion** – any matplotlib color for deletion events
- **color_fragmented** – any matplotlib color for fragmented ground truth events
- **color_fragmented_merged** – any matplotlib color for merged and fragmented ground truth events
- **color_merged** – any matplotlib color for merged ground truth events
- **color_correct** – any matplotlib color for correct events
- **color_merging** – any matplotlib color for merging detection events
- **color_merging_fragmenting** – any matplotlib color for merging and fragmenting detection events

- **color_fragmenting** – any matplotlib color for merging detection events
- **color_insertion** – any matplotlib color for insertion events

Returns matplotlib figure reference

Return type matplotlib Figure



CHAPTER 6

Utils

6.1 Standard event metrics

wardmetrics.utils.**print_standard_event_metrics**(*standard_event_results*)

Print standard precision and recall values

Examples

```
>>> print_standard_event_metrics(test_r)
Standard event results:
    precision:  0.8888888       Weighted by length:     0.9186991
    recall:     0.3333333       Weighted by length:     0.2230576
```

wardmetrics.utils.**standard_event_metrics_to_list**(*standard_event_results*)

Converting standard event metric results to a list (position of each item is fixed)

Argument: standard_event_results (dictionary): as provided by the 4th item in the results of eval_events function

Returns Item order: 1. Precision, 2. Recall 3. Length weighted precision, 4. Length weighted recall

Return type list

wardmetrics.utils.**standard_event_metrics_to_string**(*standard_event_results*, *separator=', '*, *prefix='['*, *suffix=']'*)

Converting standard event metric results to a string

Argument: standard_event_results (dictionary): as provided by the 4th item in the results of eval_events function

Keyword Arguments

- **separator** (*str*) – characters between each item

- **prefix** (*str*) – string that will be added before the line
- **suffix** (*str*) – string that will be added to the end of the line

Returns Item order: 1. Precision, 2. Recall 3. Length weighted precision, 4. Length weighted recall

Return type str

Examples

```
>>> standard_event_metrics_to_string(test_r)
[0.88888, 0.33333, 0.918, 0.22305]
>>> standard_event_metrics_to_string(test_r, separator="        ", prefix="/",␣
↪suffix="/")
/0.88888        0.33333 0.918   0.22305/
>>> standard_event_metrics_to_string(test_r, prefix="", suffix="\n")
0.88888, 0.33333, 0.918, 0.22305\n
```

6.2 Detailed event metrics

wardmetrics.utils.**print_detailed_event_metrics**(*detailed_event_results*)
Print totals for each event category

Example

```
>>> print_detailed_event_metrics(test_r)
Detailed event results:
     Actual events:
          deletions:             1       12.50% of actual events
          merged:                3       37.50% of actual events
          fragmented:            1       12.50% of actual events
          frag. and merged:      1       12.50% of actual events
          correct:               2       25.00% of actual events
    Detected events:
          insertions:            1       11.11% of detected events
          merging:               1       11.11% of detected events
          fragmenting:           4       44.44% of detected events
          frag. and merging:     1       11.11% of detected events
          correct:               2       22.22% of detected events
```

wardmetrics.utils.**detailed_event_metrics_to_list**(*detailed_event_results*)
Converting detailed event metric results to a list (position of each item is fixed)

Argument: detailed_event_results (dictionary): as provided by the 3rd item in the results of eval_events function

Returns Item order: 0. correct, 1. deletions 2. merged, 3. fragmented, 4. fragmented and merged, 5. fragmenting, 6. merging, 7. fragmenting and merging, 8. insertions, 9. total of actual events, 10. total of detected events

Return type list

wardmetrics.utils.**detailed_event_metrics_to_string**(*detailed_event_results*, *separator=', '*, *prefix='['*, *suffix=']'*)

Converting detailed event metric results to a string

Argument: detailed_event_results (dictionary): as provided by the 3rd item in the results of eval_events function

Keyword Arguments

- **separator** (*str*) – characters between each item
- **prefix** (*str*) – string that will be added before the line
- **suffix** (*str*) – string that will be added to the end of the line

Returns Item order: 0. correct, 1. deletions 2. merged, 3. fragmented, 4. fragmented and merged, 5. fragmenting, 6. merging, 7. fragmenting and merging, 8. insertions, 9. total of actual events, 10. total of detected events

Return type str

Examples

```
>>> detailed_event_metrics_to_string(test_r)
[2, 1, 3, 1, 1, 4, 1, 1, 1, 8, 9]
>>> detailed_event_metrics_to_string(test_r, separator=";", prefix="(", suffix=
↪")\n")
(2;1;3;1;1;4;1;1;1;8;9)\n
```

6.3 Detailed segment-based results

wardmetrics.utils.**print_detailed_segment_results**(*detailed_segment_results*)

Print segment length for each detailed segment category. Can be used with normed values as well.

Parameters **detailed_segment_results** (*dictionary*) – as provided by the 3rd or 4th item in the results of eval_segments function

Example

```
>>> print_detailed_segment_results(test_r)
Detailed segment results (length or frame count):
    true positive segments:             40
    true negative segments:             91
    insertion segments:                 10
    deletion segments:                  10
    fragmenting segments:               7
    merge segments:                     15
    start overfill segments:            10
    end overfill segments:              28
    start underfill segments:           13
    end underfill segments:             15
```

wardmetrics.utils.**detailed_segment_results_to_list**(*detailed_segment_results*)

Converting detailed segment results to a list (position of each item is fixed). Can be used with normed values as well.

Argument: detailed_segment_results (dictionary): as provided by the 3rd or 4th item in the results of eval_segments function

Returns Item order: 0. true posives, 1. true negatives, 2. insertions, 3. deletions, 4. fragmenting, 5. merged, 6. start overfill, 7. end overfill, 8. start underfill, 9. end underfill

Return type list

wardmetrics.utils.**detailed_segment_results_to_string**(*detailed_segment_results*, *separator=', '*, *prefix='['*, *suffix=']'*)

Converting detailed segment results to a string. Can be used with normed values as well.

Argument: detailed_segment_results (dictionary): as provided by the 3rd or 4th item in the results of eval_segments function

Keyword Arguments

- **separator** (*str*) – characters between each item
- **prefix** (*str*) – string that will be added before the line
- **suffix** (*str*) – string that will be added to the end of the line

Returns Item order: 0. true posives, 1. true negatives, 2. insertions, 3. deletions, 4. fragmenting, 5. merged, 6. start overfill, 7. end overfill, 8. start underfill, 9. end underfill

Return type str

Examples

```
>>> detailed_segment_results_to_string(test_r)
[2, 1, 3, 1, 1, 4, 1, 1, 1, 8]
>>> detailed_segment_results_to_string(test_r, separator=";", prefix="(", suffix=
↪")\n")
(2;1;3;1;1;4;1;1;1;8)\n
```

6.4 2SET segment-based metrics

wardmetrics.utils.**print_twoset_segment_metrics**(*twoset_metrics_results*)

Print 2SET metric results

Argument: twoset_metrics_results (dictionary): as provided by the 1st item in the results of eval_events function

Example

```
>>> print_twoset_segment_metrics(test_r)
2SET metrics:
    true positive rate:         0.471
    deletion rate:              0.118
    fragmenting rate:           0.082
    start underfill rate:       0.153
    end underfill rate:         0.176
```

```
    1 - false positive rate:    0.591
    insertion rate:             0.065
    merge rate:                 0.097
    start overfill rate:        0.065
    end overfill rate:          0.182
```

wardmetrics.utils.**twoset_segment_metrics_to_list**(*twoset_metrics_results*)

Converting detailed event metric results to a list (position of each item is fixed)

Argument: twoset_metrics_results (dictionary): as provided by the 1st item in the results of eval_events function

Returns Item order: 0. true positive rate, 1. deletion rate 2. fragmenting rate, 3. start underfill rate, 4. end underfill rate, 5. 1 - false positive rate, 6. insertion rate, 7. merge rate, 8. start overfill rate, 9. end overfill rate

Return type list

wardmetrics.utils.**twoset_segment_metrics_to_string**(*twoset_metrics_results*, *separator=', '*, *prefix='['*, *suffix=']'*)

Converting detailed event metric results to a string

Argument: twoset_metrics_results (dictionary): as provided by the 1st item in the results of eval_events function

Keyword Arguments

- **separator** (*str*) – characters between each item
- **prefix** (*str*) – string that will be added before the line
- **suffix** (*str*) – string that will be added to the end of the line

Returns Item order: 0. true positive rate, 1. deletion rate 2. fragmenting rate, 3. start underfill rate, 4. end underfill rate, 5. 1 - false positive rate, 6. insertion rate, 7. merge rate, 8. start overfill rate, 9. end overfill rate

Return type str

Examples

```
>>> twoset_segment_metrics_to_string(test_r)
[0.47058823529411764, 0.11764705882352941, 0.08235294117647059, 0.
↪15294117647058825, 0.17647058823529413, 0.5909090909090909, 0.06493506493506493,
↪ 0.09740259740259741, 0.06493506493506493, 0.18181818181818182]
>>> twoset_segment_metrics_to_string(test_r, separator=";", prefix="(", suffix=
↪")\n")
(0.47058823529411764;0.11764705882352941;0.08235294117647059;0.15294117647058825;
↪0.17647058823529413;0.5909090909090909;0.06493506493506493;0.09740259740259741;
↪0.06493506493506493;0.18181818181818182)\n
```

6.5 Convert frame-by-frame multiclass results to events

wardmetrics.utils.**frame_results_to_events**(*frame_results*, *frame_times=None*)

Converting frame-by-frame results into a list of events for each label. If the classifier predicts the same label in

a sequence it is considers as an event. (Filtering too short events, or merge close-by events is not included here.)

Parameters

- **frame_results** (*list or numpy array*) – list of frame class labels in an temporal order (sequential). It can contain numeric or string values e.g. `[1, 1, 0, ...]` or `['class_1', 'class_1', 'class_0', ...]`
- **frame_times** (*list or numpy array*) – list of timestamps (preferably as numeric values e.g. posix time) for each frame.

Returns list of events (tuple of start and end times/indexes) for each unique label found in the frame_results. Event start value is the first occurence of the label, event end is the time or index of the next frame after the event (also the start of the next event).

Return type dictionary

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Bibliography

[WLG11] Jamie A Ward, Paul Lukowicz, and Hans W Gellersen. Performance metrics for activity recognition. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(1):6, 2011.

Index