OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

com.aliasi.cluster

Class ClusterScore<E>

java.lang.Object
    com.aliasi.cluster.ClusterScore<E>

Type Parameters:

E - the type of objects being clustered

```
public class ClusterScore<E>
extends Object
```

A ClusterScore provides a range of cluster scoring metrics for reference partitions versus response partitions.

Traditional evaluation measures for pairs of parititions involve comparing the equivalence relations generated by the partitions pointwise. A partition defines an equivalence relation in the usual way: a pair (A,B) is in the equivalence if there is a cluster that contains both A and B. Each element is assumed to be equal to itself. A pair (A,B) is a true positive if it is an equivalence in the reference and response clustes, a false positive if it is in the response but not the reference, and so on. The resulting precision-recall statistics over the relations is reported through equivalenceEvaluation().

The scoring used for the Message Understanding Conferences is:

> mucRecall(referencePartition,responsePartition)
> = $\Sigma_{\text{c in referencePartition}}$ (size(c) - overlap(c,responsePartition))
> / $\Sigma_{\text{c in referencePartition}}$ (size(c) - 1)

where size(c) is the number of elements in the cluster c, and overlap(c,responsePartition) is the number of clusters in the response partition that intersect the cluster c. Precision is defined dually by reversing the roles of reference and response, and f-measure is defined as usual. Further details and examples can be found in:

> Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings fo the 6th Message Understanding Conference (MUC6)*. 45--52. Morgan Kaufmann.

B-cubed cluster scoring was defined as an alternative to the MUC scoring metric. There are two variants B-cubed cluster precision, both of which are weighted averages of a per-element precision score:

```
b3Precision(A,referencePartition,responsePartition)
= |cluster(responsePartition,A) INTERSECT cluster(referencePartition,A)|
/ |cluster(responsePartition,A)|
```

where cluster(partition,a) is the cluster in the partition partition containing the element a; in other words, this is A's equivalence class and contains the set of all elements equivalent to A in the partition.

For the uniform cluster method, each cluster in the reference partition is weighted equally, and each element is weighted equally within a cluster:

```
b3ClusterPrecision(referencePartition,responsePartition)
= Σa b3Precision(a,referencePartition,responsePartition)
/ (|referencePartition| * |cluster(referencePartition,a)|)
```

For the uniform element method, each element a is weighted uniformly:

```
b3ElementPrecision(ReferencePartition,ResponsePartition)
= Σa b3Precision(a,referencePartition,responsePartition) / numElements
```

where `numElements` is the total number of elements in the partitions. For both B-cubed approaches, recall is defined dually by switching the roles of reference and response, and the F_1-measure is defined in the usual way.

The B-cubed method is described in detail in:

> Bagga, Amit and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the First International Conference on Language Resources and Evaluation Workshop on Linguistic Coreference.*

As an example, consider the following two partitions:

```
reference = { {1, 2, 3, 4, 5}, {6, 7}, {8, 9, A, B, C} }
response = { { 1, 2, 3, 4, 5, 8, 9, A, B, C }, { 6, 7} }
```

which produce the following values for method calls:

Method	*Result*

`equivalenceEvaluation()`	PrecisionRecallEvaluation(54,0,50,40)
	TP=54; FN=0; FP=50; TN=40

`mucPrecision()`	0.9
`mucRecall()`	1.0
`mucF()`	0.947
`b3ElementPrecision()`	0.583
`b3ElementRecall()`	1.0
`b3ElementF()`	0.737
`b3ClusterPrecision()`	0.75
`b3ClusterRecall()`	1.0
`b3ClusterF()`	0.857

Note that there are additional scoring metrics within the `Dendrogram` class for cophenetic correlation and dendrogram-specific within-cluster scatter.

Since:

`LingPipe2.0`

Version:

`3.8`

Author:

`Bob Carpenter`

Constructor Summary

Constructors

Constructor and Description
`ClusterScore(Set<? extends Set<? extends E>> referencePartition, Set<? extends Set<? extends E>> responsePartition)` Construct a cluster score object from the specified reference and response partitions.

Method Summary

All Methods **Static Methods** **Instance Methods** **Concrete Methods**

Modifier and Type	Method and Description
`double`	`b3ClusterF()` Returns the F_1 measure of the B^3 precision and recall metrics with equal cluster weighting.
`double`	`b3ClusterPrecision()` Returns the precision as defined by B^3 metric with equal cluster weighting.
`double`	`b3ClusterRecall()` Returns the recall as defined by B^3 metric with equal cluster weighting.
`double`	`b3ElementF()` Returns the F_1 measure of the B^3 precision and recall metrics with equal element weighting.
`double`	`b3ElementPrecision()` Returns the precision as defined by B^3 metric with equal element weighting.
`double`	`b3ElementRecall()` Returns the recall as defined by B^3 metric with equal element weighting.
`PrecisionRecallEvaluation`	`equivalenceEvaluation()` Returns the precision-recall evaluation corresponding to equalities in the reference and response clusterings.
`Set<Tuple<E>>`	`falseNegatives()` Returns the set of false negative relations for this scoring.
`Set<Tuple<E>>`	`falsePositives()` Returns the set of false positive relations for this scoring.
`double`	`mucF()` Returns the F_1 measure of the MUC recall and precision.
`double`	`mucPrecision()`

	Returns the precision as defined by MUC.
`double`	`mucRecall()` Returns the recall as defined by MUC.
`static <E> double`	`scatter(Set<? extends E> cluster, Distance<? super E> distance)` Returns the scatter for the specified cluster based on the specified distance.
`String`	`toString()` Returns a string representation of the statistics for this score.
`Set<Tuple<E>>`	`truePositives()` Returns the set of true positive relations for this scoring.
`static <E> double`	`withinClusterScatter(Set<? extends Set<? extends E>> clustering, Distance<? super E> distance)` Returns the within-cluster scatter measure for the specified clustering with respect to the specified distance.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

ClusterScore

```
public ClusterScore(Set<? extends Set<? extends E>> referencePartition,
                    Set<? extends Set<? extends E>> responsePartition)
```

Construct a cluster score object from the specified reference and response partitions. A partition is a set of disjoint sets of elements. A partition defines an equivalence relation where the disjoint sets represent the equivalence classes.

The reference partition is taken to represent the "truth" or the "correct" answer, also known as the "gold standard". The response partition is the partition to evaluate against the reference partition.

If the specified partitions are not over the same sets or if the equivalence classes are not disjoint, an illegal argument exception is raised.

Parameters:

`referencePartition` - Partition of reference elements.

`responsePartition` - Partition of response elements.

Throws:

`IllegalArgumentException` - If the partitions are not valid partitions over the same set of elements.

Method Detail

equivalenceEvaluation

```
public PrecisionRecallEvaluation equivalenceEvaluation()
```

Returns the precision-recall evaluation corresponding to equalities in the reference and response clusterings.

Returns:

`The precision-recall evaluation.`

mucPrecision

```
public double mucPrecision()
```

Returns the precision as defined by MUC. See the class documentation above for definitions.

Returns:

`The precision as defined by MUC.`

mucRecall

```
public double mucRecall()
```

Returns the recall as defined by MUC. See the class documentation above for definitions.

Returns:

`The recall as defined by MUC.`

mucF

```
public double mucF()
```

Returns the F_1 measure of the MUC recall and precision. See the class documentation above for definitions.

Returns:

`The F measure as defined by MUC.`

b3ClusterPrecision

```
public double b3ClusterPrecision()
```

Returns the precision as defined by B^3 metric with equal cluster weighting. See the class documentation above for definitions.

Returns:

`The B-cubed equal cluster precision.`

b3ClusterRecall

```
public double b3ClusterRecall()
```

Returns the recall as defined by B^3 metric with equal cluster weighting. See the class documentation above for definitions.

Returns:

The B-cubed equal cluster recall.

b3ClusterF

```
public double b3ClusterF()
```

Returns the F_1 measure of the B^3 precision and recall metrics with equal cluster weighting. See the class documentation above for definitions.

Returns:

The B-cubed equal cluster F measure.

b3ElementPrecision

```
public double b3ElementPrecision()
```

Returns the precision as defined by B^3 metric with equal element weighting. See the class documentation above for definitions.

Returns:

The B-cubed equal element precision.

b3ElementRecall

```
public double b3ElementRecall()
```

Returns the recall as defined by B^3 metric with equal element weighting. See the class documentation above for definitions.

Returns:

The B-cubed equal element recall.

b3ElementF

```
public double b3ElementF()
```

Returns the F_1 measure of the B^3 precision and recall metrics with equal element weighting. See the class documentation above for definitions.

Returns:

The B-cubed equal element F measure.

truePositives

```
public Set<Tuple<E>> truePositives()
```

Returns the set of true positive relations for this scoring. Each relation is an instance of `Tuple`. These true positives will include both `(x,y)` and `(y,x)` for a true positive relation between `x` and `y`.

Returns:

The set of true positives.

falsePositives

```
public Set<Tuple<E>> falsePositives()
```

Returns the set of false positive relations for this scoring. Each relation is an instance of `Tuple`. The false positives will include both `(x,y)` and `(y,x)` for a false positive relation between `x` and `y`.

Returns:

The set of false positives.

falseNegatives

```
public Set<Tuple<E>> falseNegatives()
```

Returns the set of false negative relations for this scoring. Each relation is an instance of `Tuple`. The false negative set will include both `(x,y)` and `(y,x)` for a false negative relation between `x` and `y`.

Returns:

The set of false negatives.

toString

```
public String toString()
```

Returns a string representation of the statistics for this score. The string includes the information in all of the methods of this class: b3 scores by cluster and by element, muc scores, and the precision-recall evaluation based on equivalence.

Overrides:

`toString` in class `Object`

Returns:

String-based representation of this score.

withinClusterScatter

```
public static <E> double withinClusterScatter(Set<? extends Set<? extends E>> clustering,
                                              Distance<? super E> distance)
```

Returns the within-cluster scatter measure for the specified clustering with respect to the specified distance. The within-cluster scatter is simply the sum of the scatters for each set in the clustering; see `scatter(Set,Distance)` for a definition of scatter.

$$\begin{aligned}&\texttt{withinClusterScatter(clusters,distance)}\\&\quad = \Sigma_{\texttt{cluster in clusters}}\ \texttt{scatter(cluster,distance)}\end{aligned}$$

As the number of clusters increases, the within-cluster scatter decreases monotonically. Typically, this is used to determine how many clusters to return, by inspecting a plot of within-cluster scatter against number of clusters and looking for a "knee" in the graph.

Type Parameters:

`E` - the type of objects being clustered

Parameters:

`clustering` - Clustering to evaluate.

`distance` - Distance against which to evaluate.

Returns:

The within-cluster scatter score.

scatter

```
public static <E> double scatter(Set<? extends E> cluster,
                                 Distance<? super E> distance)
```

Returns the scatter for the specified cluster based on the specified distance. The scatter is the sum of all of the pairwise distances between elements, with each pair of elements counted once. Abusing notation to use `xs[i]` for the `i`th element returned by the set's iterator, scatter is defined by:

scatter(xs,distance)
  $= \Sigma_i \Sigma_{j < i}$ distance(xs[i],xs[j])

Note that elements are not compared to themselves. This presupposes a distance for which the distance of an element to itself is zero and which is symmetric.

Type Parameters:

`E` - the type of objects being clustered

Parameters:

`cluster` - Cluster to evaluate.

`distance` - Distance against which to evaluate.

Returns:

The total scatter for the specified set.

OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD