

---

**scandeval**

**Dan Saattrup Nielsen**

**Apr 19, 2022**



## CONTENTS:

<b>1</b>	<b>scandeval package</b>	<b>1</b>
1.1	Subpackages . . . . .	1
1.2	Submodules . . . . .	47
1.3	scandeval.benchmark module . . . . .	47
1.4	scandeval.cli module . . . . .	49
1.5	scandeval.datasets module . . . . .	49
1.6	scandeval.utils module . . . . .	55
1.7	Module contents . . . . .	56
<b>2</b>	<b>Indices and tables</b>	<b>57</b>
	<b>Python Module Index</b>	<b>59</b>
	<b>Index</b>	<b>61</b>



## SCANDEVAL PACKAGE

## 1.1 Subpackages

### 1.1.1 scandeval.benchmarks package

#### Subpackages

scandeval.benchmarks.abstract package

#### Submodules

scandeval.benchmarks.abstract.base module

Abstract base class for evaluating models

```
class scandeval.benchmarks.abstract.base.BaseBenchmark(name: str, task: str, metric_names: Dict[str, str], id2label: Optional[List[str]] = None, label_synonyms: Optional[List[List[str]] = None, evaluate_train: bool = False, cache_dir: str = '.benchmark_models', two_labels: bool = False, split_point: Optional[int] = None, verbose: bool = False)
```

Bases: abc.ABC

Abstract base class for finetuning and evaluating models.

#### Parameters

- **name** (*str*) – The name of the dataset.
- **task** (*str*) – The type of task to be benchmarked.
- **metric\_names** (*dict*) – A dictionary with the variable names of the metrics used in the dataset as keys, and a more human readable name of them as values.
- **id2label** (*list or None*) – A list of all the labels, which is used to convert indices to their labels. This will only be used if the pretrained model does not already have one. Defaults to None.
- **label\_synonyms** (*list of lists of str*) – A list of synonyms for each label. Every entry in *label\_synonyms* is a list of synonyms, where one of the synonyms is contained in *id2label*. If None then no synonyms will be used. Defaults to None.

- **evaluate\_train** (*bool*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **cache\_dir** (*str*) – Where the downloaded models will be stored. Defaults to ‘.benchmark\_models’.
- **two\_labels** (*bool*) – Whether two labels should be predicted in the dataset. If this is True then *split\_point* has to be set. Defaults to False.
- **split\_point** (*int or None*) – When there are two labels to be predicted, this is the index such that *id2label[split\_point]* contains the labels for the first label, and *id2label[split\_point]* contains the labels for the second label. Only relevant if *two\_labels* is True. Defaults to None.
- **verbose** (*bool*) – Whether to print additional output during evaluation. Defaults to False.
- **name** – The name of the dataset.
- **task** – The type of task to be benchmarked.
- **metric\_names** – The names of the metrics.
- **id2label** – A list converting indices to labels.
- **label2id** (*dict or None*) – A dictionary converting labels to indices.
- **num\_labels** (*int or None*) – The number of labels in the dataset.
- **label\_synonyms** – Synonyms of the dataset labels.
- **evaluate\_train** – Whether the training set should be evaluated.
- **cache\_dir** – Directory where models are cached.
- **two\_labels** – Whether two labels should be predicted.
- **split\_point** – Splitting point of *id2label* into labels.
- **verbose** – Whether to print additional output.

**benchmark**(*model\_id: str, progress\_bar: bool = True*) → Dict[str, dict]  
Benchmark a model.

#### Parameters

- **model\_id** (*str*) – The full HuggingFace Hub path to the pretrained transformer model. The specific model version to use can be added after the suffix ‘@’: “*model\_id@v1.0.0*”. It can be a branch name, a tag name, or a commit id (currently only supported for HuggingFace models, and it defaults to ‘main’ for latest).
- **progress\_bar** (*bool, optional*) – Whether to show a progress bar or not. Defaults to True.

**Returns** The keys in the dict are ‘raw\_metrics’ and ‘total’, with all the raw metrics in the first dictionary and the aggregated metrics in the second.

**Return type** dict

**Raises** **RuntimeError** – If the extracted framework is not recognized.

**scandeval.benchmarks.abstract.dep module**

Abstract dependency parsing benchmark

```
class scandeval.benchmarks.abstract.dep.DepBenchmark(name: str, cache_dir: str =
                                                    '.benchmark_models', evaluate_train: bool =
                                                    False, verbose: bool = False)
```

Bases: *scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark*

Abstract dependency parsing benchmark.

**Parameters**

- **name** (*str*) – The name of the dataset.
- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `'.benchmark_models'`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

### **scandeval.benchmarks.abstract.ner module**

Abstract NER tagging benchmark

```
class scandeval.benchmarks.abstract.ner.NerBenchmark(name: str, cache_dir: str =  
                                                    '.benchmark_models', evaluate_train: bool =  
                                                    False, verbose: bool = False)
```

Bases: *scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark*

Abstract NER tagging benchmark.

#### **Parameters**

- **name** (*str*) – The name of the dataset.
- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.



**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

## scandeval.benchmarks.abstract.pos module

Abstract POS tagging benchmark

```
class scandeval.benchmarks.abstract.pos.PosBenchmark(name: str, cache_dir: str =
'.benchmark_models', evaluate_train: bool =
False, verbose: bool = False)
```

Bases: *scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark*

Abstract NER tagging benchmark.

### Parameters

- **name** (*str*) – The name of the dataset.
- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.abstract.text\_classification module**

Abstract text classification benchmark

```
class scandeval.benchmarks.abstract.text_classification.TextClassificationBenchmark(name:
    str,
    id2label:
    list, label_synonyms:
    Optional[List[List[str]]]
    =
    None,
    evaluate_train:
    bool =
    False,
    cache_dir:
    str =
    'benchmark_models',
    two_labels:
    bool =
    False,
    split_point:
    Optional[int]
    =
    None,
    verbose:
    bool =
    False)
```

Bases: `scandeval.benchmarks.abstract.base.BaseBenchmark`, `abc.ABC`

Abstract text classification benchmark.

**Parameters**

- **name** (*str*) – The name of the dataset.
- **metric\_names** (*dict*) – A dictionary with the variable names of the metrics used in the dataset as keys, and a more human readable name of them as values.
- **id2label** (*list or None, optional*) – A list of all the labels, which is used to convert indices to their labels. This will only be used if the pretrained model does not already have one. Defaults to `None`.
- **label\_synonyms** (*list of lists of str or None, optional*) – A list of synonyms for each label. Every entry in `label_synonyms` is a list of synonyms, where one of the synonyms is contained in `id2label`. If `None` then no synonyms will be used. Defaults to `None`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `'benchmark_models'`.

- **two\_labels** (*bool, optional*) – Whether two labels should be predicted in the dataset. If this is True then *split\_point* has to be set. Defaults to False.
- **split\_point** (*int or None, optional*) – When there are two labels to be predicted, this is the index such that *id2label[:split\_point]* contains the labels for the first label, and *id2label[split\_point]* contains the labels for the second label. Only relevant if *two\_labels* is True. Defaults to None.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**create\_numerical\_labels**(*examples: dict, label2id: dict*) → dict

**scandeval.benchmarks.abstract.token\_classification module**

Abstract token classification benchmark

```
class scandeval.benchmarks.abstract.token_classification.TokenClassificationBenchmark(name:
    str,
    met-
    ric_names:
    Dict[str,
    str],
    id2label:
    list,
    la-
    bel_synonyms:
    Op-
    tional[List[List[str]]]
    =
    None,
    eval-
    u-
    ate_train:
    bool
    =
    False,
    cache_dir:
    str
    =
    '.bench-
    mark_models',
    two_labels:
    bool
    =
    False,
    split_point:
    Op-
    tional[int]
    =
    None,
    ver-
    bose:
    bool
    =
    False)
```

Bases: `scandeval.benchmarks.abstract.base.BaseBenchmark`, `abc.ABC`

Abstract token classification benchmark.

**Parameters**

- **name** (*str*) – The name of the dataset.
- **metric\_names** (*dict*) – A dictionary with the variable names of the metrics used in the dataset as keys, and a more human readable name of them as values.
- **id2label** (*list or None, optional*) – A list of all the labels, which is used to convert indices to their labels. This will only be used if the pretrained model does not already have one. Defaults to None.
- **label\_synonyms** (*list of lists of str or None, optional*) – A list of synonyms for each label. Every entry in *label\_synonyms* is a list of synonyms, where one of the synonyms is contained in *id2label*. If None then no synonyms will be used. Defaults to None.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **two\_labels** (*bool, optional*) – Whether two labels should be predicted in the dataset. If this is True then *split\_point* has to be set. Defaults to False.
- **split\_point** (*int or None, optional*) – When there are two labels to be predicted, this is the index such that *id2label[split\_point]* contains the labels for the first label, and *id2label[split\_point]* contains the labels for the second label. Only relevant if *two\_labels* is True. Defaults to None.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**Module contents****Submodules****scandeval.benchmarks.absabank\_imm module**

Immigration sentiment classification on the ABSABank-Imm dataset

```
class scandeval.benchmarks.absabank_imm.AbsabankImmBenchmark(cache_dir: str =
    '.benchmark_models',
    evaluate_train: bool = False,
    verbose: bool = False)
```

Bases: *scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark*

Benchmark of language models on the ABSABank-Imm dataset.

**Parameters**

- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.angry\_tweets module**

Sentiment evaluation of a language model on the AngryTweets dataset

```
class scandeval.benchmarks.angry_tweets.AngryTweetsBenchmark(cache_dir: str =  
    '.benchmark_models',  
    evaluate_train: bool = False,  
    verbose: bool = False)
```

Bases: *scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark*

Benchmark of language models on the AngryTweets dataset.

**Parameters**



- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

### scandeval.benchmarks.dalaj module

Correct spelling classification of a language model on the DaLaJ dataset

```
class scandeval.benchmarks.dalaj.DalajBenchmark(cache_dir: str = '.benchmark_models',  
                                              evaluate_train: bool = False, verbose: bool = False)
```

Bases: *scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark*

Benchmark of language models on the DaLaJ dataset.

#### Parameters

- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to `False`.

#### **name**

The name of the dataset.

**Type** str

#### **task**

The type of task to be benchmarked.

**Type** str

#### **metric\_names**

The names of the metrics.

**Type** dict

#### **id2label**

A dictionary converting indices to labels.

**Type** dict or None

#### **label2id**

A dictionary converting labels to indices.

**Type** dict or None

#### **num\_labels**

The number of labels in the dataset.

**Type** int or None

#### **label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

#### **evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

#### **cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

## scandeval.benchmarks.dane module

NER evaluation of a language model on the DaNE dataset

**class** scandeval.benchmarks.dane.DaneBenchmark(*cache\_dir: str = '.benchmark\_models', evaluate\_train: bool = False, verbose: bool = False*)

Bases: *scandeval.benchmarks.abstract.ner.NerBenchmark*

Benchmark of language models on the DaNE dataset.

### Parameters

- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.ddt\_dep module**

Dependency parsing evaluation of a language model on the DDT dataset

```
class scandeval.benchmarks.ddt_dep.DdtDepBenchmark(cache_dir: str = '.benchmark_models',  
                                                evaluate_train: bool = False, verbose: bool =  
                                                False)
```

Bases: *scandeval.benchmarks.abstract.dep.DepBenchmark*

Benchmark of language models on the dependency parsing part of the DDT.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.ddt\_pos module**

POS evaluation of a language model on the DDT dataset

```
class scandeval.benchmarks.ddt_pos.DdtPosBenchmark(cache_dir: str = '.benchmark_models',
                                                    evaluate_train: bool = False, verbose: bool =
                                                    False)
```

Bases: *scandeval.benchmarks.abstract.pos.PosBenchmark*

Benchmark of language models on the POS part of the DDT dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.

- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.dkhate module**

Hate speech classification of a language model on the DKHate dataset

```
class scandeval.benchmarks.dkhate.DKHateBenchmark(cache_dir: str = '.benchmark_models',
                                                evaluate_train: bool = False, verbose: bool =
                                                False)
```

Bases: `scandeval.benchmarks.abstract.text_classification.TextClassificationBenchmark`

Benchmark of language models on the DKHate dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.europarl module**

Sentiment evaluation of a language model on the Europarl dataset

```
class scandeval.benchmarks.europarl.EuroparlBenchmark(cache_dir: str = '.benchmark_models',  
                                                    evaluate_train: bool = False, verbose: bool =  
                                                    False)
```

Bases: *scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark*

Benchmark of language models on the Europarl dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.



**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

## scandeval.benchmarks.fdt\_dep module

Dependency parsing evaluation of a language model on the FDT dataset

```
class scandeval.benchmarks.fdt_dep.FdtDepBenchmark(cache_dir: str = '.benchmark_models',
                                                    evaluate_train: bool = False, verbose: bool =
                                                    False)
```

Bases: *scandeval.benchmarks.abstract.dep.DepBenchmark*

Benchmark of language models on the dependency parsing part of the FDT.

### Parameters

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

### name

The name of the dataset.

**Type** str

### task

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.fdt\_pos module**

POS evaluation of a language model on the FDT dataset

```
class scandeval.benchmarks.fdt_pos.FdtPosBenchmark(cache_dir: str = '.benchmark_models',  
                                                evaluate_train: bool = False, verbose: bool =  
                                                False)
```

Bases: *scandeval.benchmarks.abstract.pos.PosBenchmark*

Benchmark of language models on the POS part of the FDT dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `'benchmark_models'`.

- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.idt\_dep module**

Dependency parsing evaluation of a language model on the IDT dataset

```
class scandeval.benchmarks.idt_dep.IdtDepBenchmark(cache_dir: str = '.benchmark_models',  
                                                evaluate_train: bool = False, verbose: bool =  
                                                False)
```

Bases: *scandeval.benchmarks.abstract.dep.DepBenchmark*

Benchmark of language models on the dependency parsing part of the IDT.

**Parameters**

- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `'.benchmark_models'`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** `str`

**task**

The type of task to be benchmarked.

**Type** `str`

**metric\_names**

The names of the metrics.

**Type** `dict`

**id2label**

A dictionary converting indices to labels.

**Type** `dict` or `None`

**label2id**

A dictionary converting labels to indices.

**Type** `dict` or `None`

**num\_labels**

The number of labels in the dataset.

**Type** `int` or `None`

**label\_synonyms**

Synonyms of the dataset labels.

**Type** `list` of `lists` of `str`

**evaluate\_train**

Whether the training set should be evaluated.

**Type** `bool`

**cache\_dir**

Directory where models are cached.

**Type** `str`

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.idt\_pos module**

POS evaluation of a language model on the IDT dataset

```
class scandeval.benchmarks.idt_pos.IdtPosBenchmark(cache_dir: str = '.benchmark_models',
                                                evaluate_train: bool = False, verbose: bool =
                                                False)
```

Bases: *scandeval.benchmarks.abstract.pos.PosBenchmark*

Benchmark of language models on the POS part of the IDT dataset.

**Parameters**

- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

## scandeval.benchmarks.lcc module

Sentiment evaluation of a language model on the LCC dataset

```
class scandeval.benchmarks.lcc.LccBenchmark(cache_dir: str = '.benchmark_models', evaluate_train:  
                                           bool = False, verbose: bool = False)
```

Bases: *scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark*

Benchmark of language models on the LCC dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**  
A dictionary converting indices to labels.  
**Type** dict or None

**label2id**  
A dictionary converting labels to indices.  
**Type** dict or None

**num\_labels**  
The number of labels in the dataset.  
**Type** int or None

**label\_synonyms**  
Synonyms of the dataset labels.  
**Type** list of lists of str

**evaluate\_train**  
Whether the training set should be evaluated.  
**Type** bool

**cache\_dir**  
Directory where models are cached.  
**Type** str

**two\_labels**  
Whether two labels should be predicted.  
**Type** bool

**split\_point**  
Splitting point of *id2label* into labels.  
**Type** int or None

**verbose**  
Whether to print additional output.  
**Type** bool

### scandeval.benchmarks.mim\_gold\_ner module

NER evaluation of a language model on the MIM-GOLD-NER dataset

```
class scandeval.benchmarks.mim_gold_ner.MimGoldNerBenchmark(cache_dir: str = 'benchmark_models',
                                                         evaluate_train: bool = False, verbose:
                                                         bool = False)
```

Bases: *scandeval.benchmarks.abstract.ner.NerBenchmark*

Benchmark of language models on the MIM-GOLD-NER dataset.

#### Parameters

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.

- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool



**scandeval.benchmarks.ndt\_nb\_dep module**

Dependency parsing evaluation of a language model on the NDT-NB dataset

```
class scandeval.benchmarks.ndt_nb_dep.NdtNBDepBenchmark(cache_dir: str = '.benchmark_models',
                                                    evaluate_train: bool = False, verbose: bool
                                                    = False)
```

Bases: *scandeval.benchmarks.abstract.dep.DepBenchmark*

Benchmark of language models on the dependency parsing part of DDT-NB.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.ndt\_nb\_pos module**

POS evaluation of a language model on the Bokmål part of the NDT dataset

```
class scandeval.benchmarks.ndt_nb_pos.NdtNBPosBenchmark(cache_dir: str = '.benchmark_models',  
                                                    evaluate_train: bool = False, verbose: bool  
                                                    = False)
```

Bases: *scandeval.benchmarks.abstract.pos.PosBenchmark*

Benchmark of language models on the Bokmål POS part of the NDT dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

## scandeval.benchmarks.ndt\_nn\_dep module

Dependency parsing evaluation of a language model on the NDT-NN dataset

```
class scandeval.benchmarks.ndt_nn_dep.NdtNNDepBenchmark(cache_dir: str = '.benchmark_models',
                                                         evaluate_train: bool = False, verbose: bool
                                                         = False)
```

Bases: *scandeval.benchmarks.abstract.dep.DepBenchmark*

Benchmark of language models on the dependency parsing part of DDT-NN.

### Parameters

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

### name

The name of the dataset.

**Type** str

### task

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.ndt\_nn\_pos module**

POS evaluation of a language model on the Nynorsk part of the NDT dataset

```
class scandeval.benchmarks.ndt_nn_pos.NdtNNPosBenchmark(cache_dir: str = '.benchmark_models',  
                                                    evaluate_train: bool = False, verbose: bool  
                                                    = False)
```

Bases: *scandeval.benchmarks.abstract.pos.PosBenchmark*

Benchmark of language models on the Nynorsk POS part of the NDT dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `'benchmark_models'`.

- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.nordial module**

Dialect classification of a language model on the NorDial dataset

```
class scandeval.benchmarks.nordial.NorDialBenchmark(cache_dir: str = '.benchmark_models',  
                                                  evaluate_train: bool = False, verbose: bool =  
                                                  False)
```

Bases: `scandeval.benchmarks.abstract.text_classification.TextClassificationBenchmark`

Benchmark of language models on the NorDial dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.norec module**

Sentiment evaluation of a language model on the NoReC dataset

**class** `scandeval.benchmarks.norec.NorecBenchmark`(*cache\_dir*: str = *'benchmark\_models'*,  
*evaluate\_train*: bool = *False*, *verbose*: bool = *False*)

Bases: `scandeval.benchmarks.abstract.text_classification.TextClassificationBenchmark`

Benchmark of language models on the NoReC dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to *'benchmark\_models'*.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to *False*.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to *False*.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.norec\_fo module**

Sentiment evaluation of a language model on the NoReC-FO dataset

```
class scandeval.benchmarks.norec_fo.NorecFOBenchmark(cache_dir: str = '.benchmark_models',  
                                                    evaluate_train: bool = False, verbose: bool =  
                                                    False)
```

Bases: *scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark*

Benchmark of language models on the NoReC-FO dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.



**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

## scandeval.benchmarks.norec\_is module

Sentiment evaluation of a language model on the NoReC-IS dataset

```
class scandeval.benchmarks.norec_is.NoRecISBenchmark(cache_dir: str = 'benchmark_models',
                                                    evaluate_train: bool = False, verbose: bool =
                                                    False)
```

Bases: *scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark*

Benchmark of language models on the NoReC-IS dataset.

### Parameters

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.

- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.norne\_nb module**

NER evaluation of a language model on the Bokmål part of NorNE

```
class scandeval.benchmarks.norne_nb.NorneNBBenchmark(cache_dir: str = '.benchmark_models',
                                                    evaluate_train: bool = False, verbose: bool =
                                                    False)
```

Bases: *scandeval.benchmarks.abstract.ner.NerBenchmark*

Benchmark of language models on the Bokmål part of the NorNE dataset.

**Parameters**

- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.norne\_nn module**

NER evaluation of a language model on the Nynorsk part of NorNE

```
class scandeval.benchmarks.norne_nn.NorneNNBenchmark(cache_dir: str = '.benchmark_models',  
                                                    evaluate_train: bool = False, verbose: bool =  
                                                    False)
```

Bases: *scandeval.benchmarks.abstract.ner.NerBenchmark*

Benchmark of language models on the Nynorsk part of the NorNE dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

## scandeval.benchmarks.sdt\_dep module

Dependency parsing evaluation of a language model on the SDT dataset

```
class scandeval.benchmarks.sdt_dep.SdtDepBenchmark(cache_dir: str = '.benchmark_models',
                                                evaluate_train: bool = False, verbose: bool =
                                                False)
```

Bases: *scandeval.benchmarks.abstract.dep.DepBenchmark*

Benchmark of language models on the dependency parsing part of the SDT.

### Parameters

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

### name

The name of the dataset.

**Type** str

### task

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.sdt\_pos module**

POS evaluation of a language model on the SDT dataset

```
class scandeval.benchmarks.sdt_pos.SdtPosBenchmark(cache_dir: str = '.benchmark_models',  
                                                evaluate_train: bool = False, verbose: bool =  
                                                False)
```

Bases: *scandeval.benchmarks.abstract.pos.PosBenchmark*

Benchmark of language models on the POS part of the SDT dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `'benchmark_models'`.

- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to False.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to False.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.suc3 module**

NER evaluation of a language model on the SUC 3.0 dataset

```
class scandeval.benchmarks.suc3.Suc3Benchmark(cache_dir: str = '.benchmark_models', evaluate_train:  
                                             bool = False, verbose: bool = False)
```

Bases: `scandeval.benchmarks.abstract.ner.NerBenchmark`

Benchmark of language models on the NER part of the SUC 3.0 dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str



**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.twitter\_sent module**

Sentiment evaluation of a language model on the TwitterSent dataset

```
class scandeval.benchmarks.twitter_sent.TwitterSentBenchmark(cache_dir: str =
    '.benchmark_models',
    evaluate_train: bool = False,
    verbose: bool = False)
```

Bases: *scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark*

Benchmark of language models on the TwitterSent dataset.

**Parameters**

- **cache\_dir** (*str, optional*) – Where the downloaded models will be stored. Defaults to `'.benchmark_models'`.
- **evaluate\_train** (*bool, optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`.
- **verbose** (*bool, optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

**task**

The type of task to be benchmarked.

**Type** str

**metric\_names**

The names of the metrics.

**Type** dict

**id2label**

A dictionary converting indices to labels.

**Type** dict or None

**label2id**

A dictionary converting labels to indices.

**Type** dict or None

**num\_labels**

The number of labels in the dataset.

**Type** int or None

**label\_synonyms**

Synonyms of the dataset labels.

**Type** list of lists of str

**evaluate\_train**

Whether the training set should be evaluated.

**Type** bool

**cache\_dir**

Directory where models are cached.

**Type** str

**two\_labels**

Whether two labels should be predicted.

**Type** bool

**split\_point**

Splitting point of *id2label* into labels.

**Type** int or None

**verbose**

Whether to print additional output.

**Type** bool

**scandeval.benchmarks.wikiann\_fo module**

NER evaluation of a language model on the Faroese WikiANN dataset

```
class scandeval.benchmarks.wikiann_fo.WikiannFoBenchmark(cache_dir: str = '.benchmark_models',  
                                                    evaluate_train: bool = False, verbose:  
                                                    bool = False)
```

Bases: *scandeval.benchmarks.abstract.ner.NerBenchmark*

Benchmark of language models on the Faroese WikiANN dataset.

**Parameters**

- **cache\_dir** (*str*, *optional*) – Where the downloaded models will be stored. Defaults to `‘.benchmark_models’`.
- **evaluate\_train** (*bool*, *optional*) – Whether the models should be evaluated on the training scores. Defaults to `False`. Whether to prefer Jax for the evaluation. Defaults to `False`.
- **verbose** (*bool*, *optional*) – Whether to print additional output during evaluation. Defaults to `False`.

**name**

The name of the dataset.

**Type** str

---

<b>task</b>	The type of task to be benchmarked.
<b>Type</b>	str
<b>metric_names</b>	The names of the metrics.
<b>Type</b>	dict
<b>id2label</b>	A dictionary converting indices to labels.
<b>Type</b>	dict or None
<b>label2id</b>	A dictionary converting labels to indices.
<b>Type</b>	dict or None
<b>num_labels</b>	The number of labels in the dataset.
<b>Type</b>	int or None
<b>label_synonyms</b>	Synonyms of the dataset labels.
<b>Type</b>	list of lists of str
<b>evaluate_train</b>	Whether the training set should be evaluated.
<b>Type</b>	bool
<b>cache_dir</b>	Directory where models are cached.
<b>Type</b>	str
<b>two_labels</b>	Whether two labels should be predicted.
<b>Type</b>	bool
<b>split_point</b>	Splitting point of <i>id2label</i> into labels.
<b>Type</b>	int or None

## Module contents

### 1.2 Submodules

#### 1.3 scandeval.benchmark module

Fetches an updated list of all Scandinavian models on the HuggingFace Hub

```
class scandeval.benchmark.Benchmark(progress_bar: bool = True, save_results: bool = False, language:  
Union[str, List[str]] = ['da', 'sv', 'no', 'nb', 'nn', 'is', 'fo'], task:  
Union[str, List[str]] = 'all', evaluate_train: bool = False, verbose:  
bool = False)
```

Bases: object

Benchmarking all the Scandinavian language models.

#### Parameters

- **progress\_bar** (*bool, optional*) – Whether progress bars should be shown. Defaults to True.
- **save\_results** (*bool, optional*) – Whether to save the benchmark results to ‘scandeval\_benchmark\_results.json’. Defaults to False.
- **language** (*str or list of str, optional*) – The language codes of the languages to include in the list. Set this to ‘all’ if all languages (also non-Scandinavian) should be considered. Defaults to [‘da’, ‘sv’, ‘no’, ‘nb’, ‘nn’, ‘is’, ‘fo’].
- **task** (*str or list of str, optional*) – The tasks to consider in the list. Set this to ‘all’ if all tasks should be considered. Defaults to ‘all’.
- **evaluate\_train** (*bool, optional*) – Whether to evaluate the training set as well. Defaults to False.
- **verbose** (*bool, optional*) – Whether to output additional output. Defaults to False.

#### **progress\_bar**

Whether progress bars should be shown.

**Type** bool

#### **save\_results**

Whether to save the benchmark results.

**Type** bool

#### **language**

The languages to include in the list.

**Type** str or list of str

#### **task**

The tasks to consider in the list.

**Type** str or list of str

#### **evaluate\_train**

Whether to evaluate the training set as well.

**Type** bool

#### **verbose**

Whether to output additional output.

**Type** bool

#### **benchmark\_results**

The benchmark results.

**Type** dict

**benchmark**(*model\_id: Optional[Union[str, List[str]]] = None, dataset: Optional[Union[str, List[str]]] = None, progress\_bar: Optional[bool] = None, save\_results: Optional[bool] = None, language: Optional[Union[str, List[str]]] = None, task: Optional[Union[str, List[str]]] = None, evaluate\_train: Optional[bool] = None, verbose: Optional[bool] = None*) → Dict[str, Dict[str, dict]]

Benchmarks models on datasets.

#### Parameters

- **model\_id** (*str, list of str or None, optional*) – The model ID(s) of the models to benchmark. If None then all relevant model IDs will be benchmarked. Defaults to None.
- **dataset** (*str, list of str or None, optional*) – The datasets to benchmark on. If None then all datasets will be benchmarked. Defaults to None.
- **progress\_bar** (*bool or None, optional*) – Whether progress bars should be shown. If None then the default value from the constructor will be used. Defaults to None.
- **save\_results** (*bool or None, optional*) – Whether to save the benchmark results to ‘scandeval\_benchmark\_results.json’. If None then the default value from the constructor will be used. Defaults to None.
- **language** (*str, list of str or None, optional*) – The language codes of the languages to include in the list. Set this to ‘all’ if all languages (also non-Scandinavian) should be considered. If None then the default value from the constructor will be used. Defaults to None.
- **task** (*str, list of str or None, optional*) – The tasks to consider in the list. Set this to ‘all’ if all tasks should be considered. If None then the default value from the constructor will be used. Defaults to None.
- **evaluate\_train** (*bool or None, optional*) – Whether to evaluate the training set as well. If None then the default value from the constructor will be used. Defaults to None.
- **verbose** (*bool or None, optional*) – Whether to output additional output. If None then the default value from the constructor will be used. Defaults to None.

**Returns** A nested dictionary of the benchmark results. The keys are the names of the datasets, with values being new dictionaries having the model IDs as keys.

**Return type** dict

## 1.4 scandeval.cli module

Command-line interface for benchmarking

## 1.5 scandeval.datasets module

Functions that load datasets

`scandeval.datasets.load_absabank_imm()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the ABSAbank-Imm dataset.

**Returns** Four dataframes,  $X_{train}$ ,  $X_{test}$ ,  $y_{train}$  and  $y_{test}$ , where  $X_{train}$  and  $X_{test}$  corresponds to the feature matrices for the training and test split, respectively, and  $y_{train}$  and  $y_{test}$  contains the target vectors.

**Return type** tuple

```
scandeval.datasets.load_angry_tweets() → Tuple[pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame]
```

Load the AngryTweets dataset.

**Returns** Four dataframes,  $X_{train}$ ,  $X_{test}$ ,  $y_{train}$  and  $y_{test}$ , where  $X_{train}$  and  $X_{test}$  corresponds to the feature matrices for the training and test split, respectively, and  $y_{train}$  and  $y_{test}$  contains the target vectors.

**Return type** tuple

```
scandeval.datasets.load_dalaj() → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]
```

Load the DaLaJ dataset.

**Returns** Four dataframes,  $X_{train}$ ,  $X_{test}$ ,  $y_{train}$  and  $y_{test}$ , where  $X_{train}$  and  $X_{test}$  corresponds to the feature matrices for the training and test split, respectively, and  $y_{train}$  and  $y_{test}$  contains the target vectors.

**Return type** tuple

```
scandeval.datasets.load_dane() → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]
```

Load the the DaNE dataset.

**Returns** Four dataframes,  $X_{train}$ ,  $X_{test}$ ,  $y_{train}$  and  $y_{test}$ , where  $X_{train}$  and  $X_{test}$  corresponds to the feature matrices for the training and test split, respectively, and  $y_{train}$  and  $y_{test}$  contains the target vectors.

**Return type** tuple

```
scandeval.datasets.load_dataset(name: str) → Tuple[pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame]
```

Load a benchmark dataset.

**Parameters** **name** (*str*) – Name of the dataset.

**Returns** Four dataframes,  $X_{train}$ ,  $X_{test}$ ,  $y_{train}$  and  $y_{test}$ , where  $X_{train}$  and  $X_{test}$  corresponds to the feature matrices for the training and test split, respectively, and  $y_{train}$  and  $y_{test}$  contains the target vectors.

**Return type** tuple

**Raises** **RuntimeError** – If *name* is not a valid dataset name.

```
scandeval.datasets.load_ddt_dep() → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]
```

Load the dependency parsing part of the DDT dataset.

**Returns** Four dataframes,  $X_{train}$ ,  $X_{test}$ ,  $y_{train}$  and  $y_{test}$ , where  $X_{train}$  and  $X_{test}$  corresponds to the feature matrices for the training and test split, respectively, and  $y_{train}$  and  $y_{test}$  contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_ddt_pos()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the POS part of the DDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_dkhate()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the DKHate dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_europarl()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the Europarl dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_fdt_dep()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the dependency parsing part of the FDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_fdt_pos()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the POS part of the FDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_idt_dep()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the dependency parsing part of the IDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_idt_pos()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the POS part of the IDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_lcc()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the LCC dataset.

This dataset is the concatenation of the LCC1 and LCC2 datasets.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_mim_gold_ner()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the the MIM-GOLD-NER dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_ndt_nb_dep()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the Bokmål POS part of the NDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_ndt_nb_pos()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the Bokmål POS part of the NDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_ndt_nn_dep()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the Nynorsk POS part of the NDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple



`scandeval.datasets.load_ndt_nn_pos()` → Tuple[pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame]

Load the Nynorsk POS part of the NDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_nordial()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the Bokmål/Nynorsk part of the NorDial dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_norec()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the NoReC dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_norec_fo()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the NoReC-FO dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_norec_is()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the NoReC-IS dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_norne_nb()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame,  
pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the the Bokmål part of the NorNE dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_norne_nn()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the the Nynorsk part of the NorNE dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_sdt_dep()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the dependency parsing part of the SDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_sdt_pos()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the POS part of the SDT dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_suc3()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the SUC 3.0 dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_twitter_sent()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the TwitterSent dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

`scandeval.datasets.load_wikiann_fo()` → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Load the the Faroese WikiANN dataset.

**Returns** Four dataframes, *X\_train*, *X\_test*, *y\_train* and *y\_test*, where *X\_train* and *X\_test* corresponds to the feature matrices for the training and test split, respectively, and *y\_train* and *y\_test* contains the target vectors.

**Return type** tuple

## 1.6 scandeval.utils module

Utility functions to be used in other scripts

**class** `scandeval.utils.DocInherit`(*mthd: Callable*)

Bases: `object`

Docstring inheriting method descriptor.

The class itself is also used as a decorator.

**get\_no\_inst**(*cls*)

**get\_with\_inst**(*obj, cls*)

**use\_parent\_doc**(*func, source*)

**exception** `scandeval.utils.InvalidBenchmark`(*message: str = 'This model cannot be benchmarked on the given dataset.'*)

Bases: `Exception`

**class** `scandeval.utils.NeverLeaveProgressCallback`

Bases: `transformers.trainer_callback.ProgressCallback`

Progress callback which never leaves the progress bar

**on\_prediction\_step**(*args, state, control, eval\_dataloader=None, \*\*kwargs*)

Event called after a prediction step.

**on\_train\_begin**(*args, state, control, \*\*kwargs*)

Event called at the beginning of training.

**class** `scandeval.utils.TwoLabelTrainer`(*split\_point: int, \*\*kwargs*)

Bases: `transformers.trainer.Trainer`

Trainer class which deals with two labels.

**compute\_loss**(*model, inputs, return\_outputs=False*)

How the loss is computed by Trainer. By default, all models return the loss in the first element.

Subclass and override for custom behavior.

`scandeval.utils.block_terminal_output()`

Blocks libraries from writing output to the terminal

`scandeval.utils.doc_inherit`

alias of `scandeval.utils.DocInherit`

`scandeval.utils.get_all_datasets()` → list

Load a list of all datasets.

**Returns** First entry in each tuple is the short name of the dataset, second entry the long name, third entry the benchmark class and fourth entry the loading function.

**Return type** list of tuples

`scandeval.utils.is_module_installed(module: str)` → bool

Check if a module is installed.

**Parameters** `module` (*str*) – The name of the module.

**Returns** Whether the module is installed or not.

**Return type** bool

## 1.7 Module contents

## INDICES AND TABLES

- genindex
- modindex
- search



## PYTHON MODULE INDEX

### S

scandeval, 56  
scandeval.benchmark, 47  
scandeval.benchmarks, 47  
scandeval.benchmarks.absabank\_imm, 11  
scandeval.benchmarks.abstract, 11  
scandeval.benchmarks.abstract.base, 1  
scandeval.benchmarks.abstract.dep, 3  
scandeval.benchmarks.abstract.ner, 4  
scandeval.benchmarks.abstract.pos, 5  
scandeval.benchmarks.abstract.text\_classification,  
7  
scandeval.benchmarks.abstract.token\_classification,  
9  
scandeval.benchmarks.angry\_tweets, 12  
scandeval.benchmarks.dalaj, 14  
scandeval.benchmarks.dane, 15  
scandeval.benchmarks.ddt\_dep, 16  
scandeval.benchmarks.ddt\_pos, 17  
scandeval.benchmarks.dkhate, 19  
scandeval.benchmarks.europarl, 20  
scandeval.benchmarks.fdt\_dep, 21  
scandeval.benchmarks.fdt\_pos, 22  
scandeval.benchmarks.idt\_dep, 24  
scandeval.benchmarks.idt\_pos, 25  
scandeval.benchmarks.lcc, 26  
scandeval.benchmarks.mim\_gold\_ner, 27  
scandeval.benchmarks.ndt\_nb\_dep, 29  
scandeval.benchmarks.ndt\_nb\_pos, 30  
scandeval.benchmarks.ndt\_nn\_dep, 31  
scandeval.benchmarks.ndt\_nn\_pos, 32  
scandeval.benchmarks.nordial, 34  
scandeval.benchmarks.norec, 35  
scandeval.benchmarks.norec\_fo, 36  
scandeval.benchmarks.norec\_is, 37  
scandeval.benchmarks.norne\_nb, 39  
scandeval.benchmarks.norne\_nn, 40  
scandeval.benchmarks.sdt\_dep, 41  
scandeval.benchmarks.sdt\_pos, 42  
scandeval.benchmarks.suc3, 44  
scandeval.benchmarks.twitter\_sent, 45  
scandeval.benchmarks.wikiann\_fo, 46  
scandeval.cli, 49  
scandeval.datasets, 49  
scandeval.utils, 55





## INDEX

### A

**AbsabankImmBenchmark** (class in *scandeval.benchmarks.absabank\_imm*), 11  
**AngryTweetsBenchmark** (class in *scandeval.benchmarks.angry\_tweets*), 12

### B

**BaseBenchmark** (class in *scandeval.benchmarks.abstract.base*), 1  
**Benchmark** (class in *scandeval.benchmark*), 47  
**benchmark()** (*scandeval.benchmark.Benchmark* method), 48  
**benchmark()** (*scandeval.benchmarks.abstract.base.BaseBenchmark* method), 2  
**benchmark\_results** (*scandeval.benchmark.Benchmark* attribute), 48  
**block\_terminal\_output()** (in module *scandeval.utils*), 55

### C

**cache\_dir** (*scandeval.benchmarks.absabank\_imm.AbsabankImmBenchmark* attribute), 12  
**cache\_dir** (*scandeval.benchmarks.abstract.dep.DepBenchmark* attribute), 3  
**cache\_dir** (*scandeval.benchmarks.abstract.ner.NerBenchmark* attribute), 5  
**cache\_dir** (*scandeval.benchmarks.abstract.pos.PosBenchmark* attribute), 6  
**cache\_dir** (*scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark* attribute), 8  
**cache\_dir** (*scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark* attribute), 11  
**cache\_dir** (*scandeval.benchmarks.angry\_tweets.AngryTweetsBenchmark* attribute), 13  
**cache\_dir** (*scandeval.benchmarks.dalaj.DalajBenchmark* attribute), 14  
**cache\_dir** (*scandeval.benchmarks.dane.DaneBenchmark* attribute), 16  
**cache\_dir** (*scandeval.benchmarks.ddt\_dep.DdtDepBenchmark* attribute), 17  
**cache\_dir** (*scandeval.benchmarks.ddt\_pos.DdtPosBenchmark* attribute), 18  
**cache\_dir** (*scandeval.benchmarks.dkhate.DkHateBenchmark* attribute), 19  
**cache\_dir** (*scandeval.benchmarks.europarl.EuroparlBenchmark* attribute), 21  
**cache\_dir** (*scandeval.benchmarks.fdt\_dep.FdtDepBenchmark* attribute), 22  
**cache\_dir** (*scandeval.benchmarks.fdt\_pos.FdtPosBenchmark* attribute), 23  
**cache\_dir** (*scandeval.benchmarks.idt\_dep.IdtDepBenchmark* attribute), 24  
**cache\_dir** (*scandeval.benchmarks.idt\_pos.IdtPosBenchmark* attribute), 26  
**cache\_dir** (*scandeval.benchmarks.lcc.LccBenchmark* attribute), 27  
**cache\_dir** (*scandeval.benchmarks.mim\_gold\_ner.MimGoldNerBenchmark* attribute), 28  
**cache\_dir** (*scandeval.benchmarks.ndt\_nb\_dep.NdtNBDepBenchmark* attribute), 29  
**cache\_dir** (*scandeval.benchmarks.ndt\_nb\_pos.NdtNBPosBenchmark* attribute), 31  
**cache\_dir** (*scandeval.benchmarks.ndt\_nn\_dep.NdtNNDepBenchmark* attribute), 32  
**cache\_dir** (*scandeval.benchmarks.ndt\_nn\_pos.NdtNNPosBenchmark* attribute), 33  
**cache\_dir** (*scandeval.benchmarks.nordial.NorDialBenchmark* attribute), 34  
**cache\_dir** (*scandeval.benchmarks.norec.NorecBenchmark* attribute), 36  
**cache\_dir** (*scandeval.benchmarks.norec\_fo.NorecFOBenchmark* attribute), 37  
**cache\_dir** (*scandeval.benchmarks.norec\_is.NorecISBenchmark* attribute), 38  
**cache\_dir** (*scandeval.benchmarks.norne\_nb.NorneNBBenchmark* attribute), 39  
**cache\_dir** (*scandeval.benchmarks.norne\_nn.NorneNNBenchmark* attribute), 41  
**cache\_dir** (*scandeval.benchmarks.sdt\_dep.SdtDepBenchmark* attribute), 42  
**cache\_dir** (*scandeval.benchmarks.sdt\_pos.SdtPosBenchmark* attribute), 43

cache\_dir(*scandeval.benchmarks.suc3.Suc3Benchmark* attribute), 44  
 cache\_dir(*scandeval.benchmarks.twitter\_sent.TwitterSentBenchmark* attribute), 46  
 cache\_dir(*scandeval.benchmarks.wikiann\_fo.WikiannFoBenchmark* attribute), 47  
 compute\_loss() (*scandeval.utils.TwoLabelTrainer* method), 55  
 create\_numerical\_labels() (*scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark* method), 9

## D

DalajBenchmark (class in *scandeval.benchmarks.dalaj*), 14  
 DaneBenchmark (class in *scandeval.benchmarks.dane*), 15  
 DdtDepBenchmark (class in *scandeval.benchmarks.ddt\_dep*), 16  
 DdtPosBenchmark (class in *scandeval.benchmarks.ddt\_pos*), 17  
 DepBenchmark (class in *scandeval.benchmarks.abstract.dep*), 3  
 DkHateBenchmark (class in *scandeval.benchmarks.dkhate*), 19  
 doc\_inherit (in module *scandeval.utils*), 55  
 DocInherit (class in *scandeval.utils*), 55

## E

EuroparlBenchmark (class in *scandeval.benchmarks.europarl*), 20  
 evaluate\_train(*scandeval.benchmark.Benchmark* attribute), 48  
 evaluate\_train(*scandeval.benchmarks.absabank\_imm.AbsabankImmBenchmark* attribute), 12  
 evaluate\_train(*scandeval.benchmarks.abstract.dep.DepBenchmark* attribute), 3  
 evaluate\_train(*scandeval.benchmarks.abstract.ner.NerBenchmark* attribute), 5  
 evaluate\_train(*scandeval.benchmarks.abstract.pos.PosBenchmark* attribute), 6  
 evaluate\_train(*scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark* attribute), 8  
 evaluate\_train(*scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark* attribute), 10  
 evaluate\_train(*scandeval.benchmarks.angry\_tweets.AngryTweetsBenchmark* attribute), 13  
 evaluate\_train(*scandeval.benchmarks.dalaj.DalajBenchmark* attribute), 14  
 evaluate\_train(*scandeval.benchmarks.dane.DaneBenchmark* attribute), 16  
 evaluate\_train(*scandeval.benchmarks.ddt\_dep.DdtDepBenchmark* attribute), 17  
 evaluate\_train(*scandeval.benchmarks.ddt\_pos.DdtPosBenchmark* attribute), 18  
 evaluate\_train(*scandeval.benchmarks.dkhate.DkHateBenchmark* attribute), 19  
 evaluate\_train(*scandeval.benchmarks.europarl.EuroparlBenchmark* attribute), 21  
 evaluate\_train(*scandeval.benchmarks.fdt\_dep.FdtDepBenchmark* attribute), 22  
 evaluate\_train(*scandeval.benchmarks.fdt\_pos.FdtPosBenchmark* attribute), 23  
 evaluate\_train(*scandeval.benchmarks.idt\_dep.IdtDepBenchmark* attribute), 24  
 evaluate\_train(*scandeval.benchmarks.idt\_pos.IdtPosBenchmark* attribute), 26  
 evaluate\_train(*scandeval.benchmarks.lcc.LccBenchmark* attribute), 27  
 evaluate\_train(*scandeval.benchmarks.mim\_gold\_ner.MimGoldNerBenchmark* attribute), 28  
 evaluate\_train(*scandeval.benchmarks.ndt\_nb\_dep.NdtNBDepBenchmark* attribute), 29  
 evaluate\_train(*scandeval.benchmarks.ndt\_nb\_pos.NdtNBPosBenchmark* attribute), 31  
 evaluate\_train(*scandeval.benchmarks.ndt\_nn\_dep.NdtNNDepBenchmark* attribute), 32  
 evaluate\_train(*scandeval.benchmarks.ndt\_nn\_pos.NdtNNPosBenchmark* attribute), 33  
 evaluate\_train(*scandeval.benchmarks.nordial.NorDialBenchmark* attribute), 34  
 evaluate\_train(*scandeval.benchmarks.norec.NorecBenchmark* attribute), 36

- `evaluate_train` (`scandeval.benchmarks.norec_fo.NorecFOBenchmark` attribute), 37
- `evaluate_train` (`scandeval.benchmarks.norec_is.NorecISBenchmark` attribute), 38
- `evaluate_train` (`scandeval.benchmarks.norne_nb.NorneNBBenchmark` attribute), 39
- `evaluate_train` (`scandeval.benchmarks.norne_nn.NorneNNBenchmark` attribute), 41
- `evaluate_train` (`scandeval.benchmarks.sdt_dep.SdtDepBenchmark` attribute), 42
- `evaluate_train` (`scandeval.benchmarks.sdt_pos.SdtPosBenchmark` attribute), 43
- `evaluate_train` (`scandeval.benchmarks.suc3.Suc3Benchmark` attribute), 44
- `evaluate_train` (`scandeval.benchmarks.twitter_sent.TwitterSentBenchmark` attribute), 46
- `evaluate_train` (`scandeval.benchmarks.wikiann_fo.WikiannFoBenchmark` attribute), 47
- ## F
- `FdtDepBenchmark` (class in `scandeval.benchmarks.fdt_dep`), 21
- `FdtPosBenchmark` (class in `scandeval.benchmarks.fdt_pos`), 22
- ## G
- `get_all_datasets()` (in module `scandeval.utils`), 55
- `get_no_inst()` (`scandeval.utils.DocInherit` method), 55
- `get_with_inst()` (`scandeval.utils.DocInherit` method), 55
- ## I
- `id2label` (`scandeval.benchmarks.absabank_imm.AbsabankImmBenchmark` attribute), 12
- `id2label` (`scandeval.benchmarks.abstract.dep.DepBenchmark` attribute), 3
- `id2label` (`scandeval.benchmarks.abstract.ner.NerBenchmark` attribute), 4
- `id2label` (`scandeval.benchmarks.abstract.pos.PosBenchmark` attribute), 6
- `id2label` (`scandeval.benchmarks.abstract.text_classification.TextClassificationBenchmark` attribute), 8
- `id2label` (`scandeval.benchmarks.abstract.token_classification.TokenClassificationBenchmark` attribute), 10
- `id2label` (`scandeval.benchmarks.angry_tweets.AngryTweetsBenchmark` attribute), 13
- `id2label` (`scandeval.benchmarks.dalaj.DalajBenchmark` attribute), 14
- `id2label` (`scandeval.benchmarks.dane.DaneBenchmark` attribute), 15
- `id2label` (`scandeval.benchmarks.ddt_dep.DdtDepBenchmark` attribute), 17
- `id2label` (`scandeval.benchmarks.ddt_pos.DdtPosBenchmark` attribute), 18
- `id2label` (`scandeval.benchmarks.dkhate.DkHateBenchmark` attribute), 19
- `id2label` (`scandeval.benchmarks.europarl.EuroparlBenchmark` attribute), 20
- `id2label` (`scandeval.benchmarks.fdt_dep.FdtDepBenchmark` attribute), 22
- `id2label` (`scandeval.benchmarks.fdt_pos.FdtPosBenchmark` attribute), 23
- `id2label` (`scandeval.benchmarks.idt_dep.IdtDepBenchmark` attribute), 24
- `id2label` (`scandeval.benchmarks.idt_pos.IdtPosBenchmark` attribute), 25
- `id2label` (`scandeval.benchmarks.lcc.LccBenchmark` attribute), 27
- `id2label` (`scandeval.benchmarks.mim_gold_ner.MimGoldNerBenchmark` attribute), 28
- `id2label` (`scandeval.benchmarks.ndt_nb_dep.NdtNBDepBenchmark` attribute), 29
- `id2label` (`scandeval.benchmarks.ndt_nb_pos.NdtNBPosBenchmark` attribute), 30
- `id2label` (`scandeval.benchmarks.ndt_nn_dep.NdtNNDepBenchmark` attribute), 32
- `id2label` (`scandeval.benchmarks.ndt_nn_pos.NdtNNPosBenchmark` attribute), 33
- `id2label` (`scandeval.benchmarks.nordial.NorDialBenchmark` attribute), 34
- `id2label` (`scandeval.benchmarks.norec.NorecBenchmark` attribute), 35
- `id2label` (`scandeval.benchmarks.norec_fo.NorecFOBenchmark` attribute), 37
- `id2label` (`scandeval.benchmarks.norec_is.NorecISBenchmark` attribute), 38
- `id2label` (`scandeval.benchmarks.norne_nb.NorneNBBenchmark` attribute), 39
- `id2label` (`scandeval.benchmarks.norne_nn.NorneNNBenchmark` attribute), 40
- `id2label` (`scandeval.benchmarks.sdt_dep.SdtDepBenchmark` attribute), 42
- `id2label` (`scandeval.benchmarks.sdt_pos.SdtPosBenchmark` attribute), 43
- `id2label` (`scandeval.benchmarks.suc3.Suc3Benchmark` attribute), 44
- `id2label` (`scandeval.benchmarks.twitter_sent.TwitterSentBenchmark` attribute), 45

- id2label (*scandeval.benchmarks.wikiann\_fo.WikiannFoBenchmark* attribute), 47
- IdtDepBenchmark (class in *scandeval.benchmarks.idt\_dep*), 24
- IdtPosBenchmark (class in *scandeval.benchmarks.idt\_pos*), 25
- InvalidBenchmark, 55
- is\_module\_installed() (in module *scandeval.utils*), 55
- ## L
- label2id (*scandeval.benchmarks.absabank\_imm.AbsabankImmBenchmark* attribute), 12
- label2id (*scandeval.benchmarks.abstract.dep.DepBenchmark* attribute), 3
- label2id (*scandeval.benchmarks.abstract.ner.NerBenchmark* attribute), 4
- label2id (*scandeval.benchmarks.abstract.pos.PosBenchmark* attribute), 6
- label2id (*scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark* attribute), 8
- label2id (*scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark* attribute), 10
- label2id (*scandeval.benchmarks.angry\_tweets.AngryTweetsBenchmark* attribute), 13
- label2id (*scandeval.benchmarks.dalaj.DalajBenchmark* attribute), 14
- label2id (*scandeval.benchmarks.dane.DaneBenchmark* attribute), 15
- label2id (*scandeval.benchmarks.ddt\_dep.DdtDepBenchmark* attribute), 17
- label2id (*scandeval.benchmarks.ddt\_pos.DdtPosBenchmark* attribute), 18
- label2id (*scandeval.benchmarks.dkhate.DkHateBenchmark* attribute), 19
- label2id (*scandeval.benchmarks.euoparl.EuroparlBenchmark* attribute), 20
- label2id (*scandeval.benchmarks.fdt\_dep.FdtDepBenchmark* attribute), 22
- label2id (*scandeval.benchmarks.fdt\_pos.FdtPosBenchmark* attribute), 23
- label2id (*scandeval.benchmarks.idt\_dep.IdtDepBenchmark* attribute), 24
- label2id (*scandeval.benchmarks.idt\_pos.IdtPosBenchmark* attribute), 25
- label2id (*scandeval.benchmarks.lcc.LccBenchmark* attribute), 27
- label2id (*scandeval.benchmarks.mim\_gold\_ner.MimGoldNerBenchmark* attribute), 28
- label2id (*scandeval.benchmarks.ndt\_nb\_dep.NdtNBDepBenchmark* attribute), 29
- label2id (*scandeval.benchmarks.ndt\_nb\_pos.NdtNBPosBenchmark* attribute), 30
- label2id (*scandeval.benchmarks.ndt\_nn\_dep.NdtNNDepBenchmark* attribute), 32
- label2id (*scandeval.benchmarks.ndt\_nn\_pos.NdtNNPosBenchmark* attribute), 33
- label2id (*scandeval.benchmarks.nordial.NorDialBenchmark* attribute), 34
- label2id (*scandeval.benchmarks.norec.NorecBenchmark* attribute), 35
- label2id (*scandeval.benchmarks.norec\_fo.NorecFOBenchmark* attribute), 37
- label2id (*scandeval.benchmarks.norec\_is.NorecISBenchmark* attribute), 38
- label2id (*scandeval.benchmarks.norne\_nb.NorneNBBenchmark* attribute), 39
- label2id (*scandeval.benchmarks.norne\_nn.NorneNNBenchmark* attribute), 40
- label2id (*scandeval.benchmarks.sdt\_dep.SdtDepBenchmark* attribute), 42
- label2id (*scandeval.benchmarks.sdt\_pos.SdtPosBenchmark* attribute), 42
- label2id (*scandeval.benchmarks.suc3.Suc3Benchmark* attribute), 44
- label2id (*scandeval.benchmarks.twitter\_sent.TwitterSentBenchmark* attribute), 45
- label2id (*scandeval.benchmarks.wikiann\_fo.WikiannFoBenchmark* attribute), 47
- label\_synonyms (*scandeval.benchmarks.absabank\_imm.AbsabankImmBenchmark* attribute), 12
- label\_synonyms (*scandeval.benchmarks.abstract.dep.DepBenchmark* attribute), 3
- label\_synonyms (*scandeval.benchmarks.abstract.ner.NerBenchmark* attribute), 5
- label\_synonyms (*scandeval.benchmarks.abstract.pos.PosBenchmark* attribute), 6
- label\_synonyms (*scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark* attribute), 8
- label\_synonyms (*scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark* attribute), 10
- label\_synonyms (*scandeval.benchmarks.angry\_tweets.AngryTweetsBenchmark* attribute), 13
- label\_synonyms (*scandeval.benchmarks.dalaj.DalajBenchmark* attribute), 14
- label\_synonyms (*scandeval.benchmarks.dane.DaneBenchmark* attribute), 16
- label\_synonyms (*scandeval.benchmarks.dkhate.DkHateBenchmark* attribute), 19
- label\_synonyms (*scandeval.benchmarks.euoparl.EuroparlBenchmark* attribute), 20
- label\_synonyms (*scandeval.benchmarks.fdt\_dep.FdtDepBenchmark* attribute), 22
- label\_synonyms (*scandeval.benchmarks.fdt\_pos.FdtPosBenchmark* attribute), 23
- label\_synonyms (*scandeval.benchmarks.idt\_dep.IdtDepBenchmark* attribute), 24
- label\_synonyms (*scandeval.benchmarks.idt\_pos.IdtPosBenchmark* attribute), 25
- label\_synonyms (*scandeval.benchmarks.lcc.LccBenchmark* attribute), 27
- label\_synonyms (*scandeval.benchmarks.mim\_gold\_ner.MimGoldNerBenchmark* attribute), 28
- label\_synonyms (*scandeval.benchmarks.ndt\_nb\_dep.NdtNBDepBenchmark* attribute), 29
- label\_synonyms (*scandeval.benchmarks.ndt\_nb\_pos.NdtNBPosBenchmark* attribute), 30
- label\_synonyms (*scandeval.benchmarks.ndt\_nn\_dep.NdtNNDepBenchmark* attribute), 32
- label\_synonyms (*scandeval.benchmarks.ndt\_nn\_pos.NdtNNPosBenchmark* attribute), 33
- label\_synonyms (*scandeval.benchmarks.nordial.NorDialBenchmark* attribute), 34
- label\_synonyms (*scandeval.benchmarks.norec.NorecBenchmark* attribute), 35
- label\_synonyms (*scandeval.benchmarks.norec\_fo.NorecFOBenchmark* attribute), 37
- label\_synonyms (*scandeval.benchmarks.norec\_is.NorecISBenchmark* attribute), 38
- label\_synonyms (*scandeval.benchmarks.norne\_nb.NorneNBBenchmark* attribute), 39
- label\_synonyms (*scandeval.benchmarks.norne\_nn.NorneNNBenchmark* attribute), 40
- label\_synonyms (*scandeval.benchmarks.sdt\_dep.SdtDepBenchmark* attribute), 42
- label\_synonyms (*scandeval.benchmarks.sdt\_pos.SdtPosBenchmark* attribute), 42
- label\_synonyms (*scandeval.benchmarks.suc3.Suc3Benchmark* attribute), 44
- label\_synonyms (*scandeval.benchmarks.twitter\_sent.TwitterSentBenchmark* attribute), 45
- label\_synonyms (*scandeval.benchmarks.wikiann\_fo.WikiannFoBenchmark* attribute), 47

<code>val.benchmarks.ddt_dep.DdtDepBenchmark</code> <i>attribute</i> ), 17	<code>val.benchmarks.norne_nb.NorneNBBenchmark</code> <i>attribute</i> ), 39
<code>label_synonyms</code> ( <i>scandeval.benchmarks.ddt_pos.DdtPosBenchmark</i> <i>attribute</i> ), 18	<code>label_synonyms</code> ( <i>scandeval.benchmarks.norne_nn.NorneNNBenchmark</i> <i>attribute</i> ), 41
<code>label_synonyms</code> ( <i>scandeval.benchmarks.dkhate.DkHateBenchmark</i> <i>attribute</i> ), 19	<code>label_synonyms</code> ( <i>scandeval.benchmarks.sdt_dep.SdtDepBenchmark</i> <i>attribute</i> ), 42
<code>label_synonyms</code> ( <i>scandeval.benchmarks.europarl.EuroparlBenchmark</i> <i>attribute</i> ), 21	<code>label_synonyms</code> ( <i>scandeval.benchmarks.sdt_pos.SdtPosBenchmark</i> <i>attribute</i> ), 43
<code>label_synonyms</code> ( <i>scandeval.benchmarks.fdt_dep.FdtDepBenchmark</i> <i>attribute</i> ), 22	<code>label_synonyms</code> ( <i>scandeval.benchmarks.suc3.Suc3Benchmark</i> <i>attribute</i> ), 44
<code>label_synonyms</code> ( <i>scandeval.benchmarks.fdt_pos.FdtPosBenchmark</i> <i>attribute</i> ), 23	<code>label_synonyms</code> ( <i>scandeval.benchmarks.twitter_sent.TwitterSentBenchmark</i> <i>attribute</i> ), 46
<code>label_synonyms</code> ( <i>scandeval.benchmarks.idt_dep.IdtDepBenchmark</i> <i>attribute</i> ), 24	<code>label_synonyms</code> ( <i>scandeval.benchmarks.wikiann_fo.WikiannFoBenchmark</i> <i>attribute</i> ), 47
<code>label_synonyms</code> ( <i>scandeval.benchmarks.idt_pos.IdtPosBenchmark</i> <i>attribute</i> ), 26	<code>language</code> ( <i>scandeval.benchmark.Benchmark</i> <i>attribute</i> ), 48
<code>label_synonyms</code> ( <i>scandeval.benchmarks.lcc.LccBenchmark</i> <i>attribute</i> ), 27	<code>LccBenchmark</code> ( <i>class in scandeval.benchmarks.lcc</i> ), 26
<code>label_synonyms</code> ( <i>scandeval.benchmarks.mim_gold_ner.MimGoldNerBenchmark</i> <i>attribute</i> ), 28	<code>load_absabank_imm()</code> ( <i>in module scandeval.datasets</i> ), 49
<code>label_synonyms</code> ( <i>scandeval.benchmarks.ndt_nb_dep.NdtNBDepBenchmark</i> <i>attribute</i> ), 29	<code>load_angry_tweets()</code> ( <i>in module scandeval.datasets</i> ), 50
<code>label_synonyms</code> ( <i>scandeval.benchmarks.ndt_nb_pos.NdtNBPosBenchmark</i> <i>attribute</i> ), 31	<code>load_dalaj()</code> ( <i>in module scandeval.datasets</i> ), 50
<code>label_synonyms</code> ( <i>scandeval.benchmarks.ndt_nn_dep.NdtNNDepBenchmark</i> <i>attribute</i> ), 32	<code>load_dane()</code> ( <i>in module scandeval.datasets</i> ), 50
<code>label_synonyms</code> ( <i>scandeval.benchmarks.ndt_nn_pos.NdtNNPosBenchmark</i> <i>attribute</i> ), 33	<code>load_dataset()</code> ( <i>in module scandeval.datasets</i> ), 50
<code>label_synonyms</code> ( <i>scandeval.benchmarks.nordial.NorDialBenchmark</i> <i>attribute</i> ), 34	<code>load_ddt_dep()</code> ( <i>in module scandeval.datasets</i> ), 50
<code>label_synonyms</code> ( <i>scandeval.benchmarks.norec.NorecBenchmark</i> <i>at-</i> <i>tribute</i> ), 35	<code>load_ddt_pos()</code> ( <i>in module scandeval.datasets</i> ), 50
<code>label_synonyms</code> ( <i>scandeval.benchmarks.norec_fo.NorecFOBenchmark</i> <i>attribute</i> ), 37	<code>load_dkhate()</code> ( <i>in module scandeval.datasets</i> ), 51
<code>label_synonyms</code> ( <i>scandeval.benchmarks.norec_is.NorecISBenchmark</i> <i>attribute</i> ), 38	<code>load_europarl()</code> ( <i>in module scandeval.datasets</i> ), 51
<code>label_synonyms</code> ( <i>scandeval.benchmarks.norne_nb.NorneNBBenchmark</i> <i>attribute</i> ), 39	<code>load_fdt_dep()</code> ( <i>in module scandeval.datasets</i> ), 51
<code>label_synonyms</code> ( <i>scandeval.benchmarks.norne_nn.NorneNNBenchmark</i> <i>attribute</i> ), 41	<code>load_fdt_pos()</code> ( <i>in module scandeval.datasets</i> ), 51
<code>label_synonyms</code> ( <i>scandeval.benchmarks.sdt_dep.SdtDepBenchmark</i> <i>attribute</i> ), 42	<code>load_idt_dep()</code> ( <i>in module scandeval.datasets</i> ), 51
<code>label_synonyms</code> ( <i>scandeval.benchmarks.sdt_pos.SdtPosBenchmark</i> <i>attribute</i> ), 43	<code>load_idt_pos()</code> ( <i>in module scandeval.datasets</i> ), 51
<code>label_synonyms</code> ( <i>scandeval.benchmarks.suc3.Suc3Benchmark</i> <i>attribute</i> ), 44	<code>load_lcc()</code> ( <i>in module scandeval.datasets</i> ), 52
<code>label_synonyms</code> ( <i>scandeval.benchmarks.twitter_sent.TwitterSentBenchmark</i> <i>attribute</i> ), 46	<code>load_mim_gold_ner()</code> ( <i>in module scandeval.datasets</i> ), 52
<code>label_synonyms</code> ( <i>scandeval.benchmarks.wikiann_fo.WikiannFoBenchmark</i> <i>attribute</i> ), 47	<code>load_ndt_nb_dep()</code> ( <i>in module scandeval.datasets</i> ), 52
<code>language</code> ( <i>scandeval.benchmark.Benchmark</i> <i>attribute</i> ), 48	<code>load_ndt_nb_pos()</code> ( <i>in module scandeval.datasets</i> ), 52
<code>LccBenchmark</code> ( <i>class in scandeval.benchmarks.lcc</i> ), 26	<code>load_ndt_nn_dep()</code> ( <i>in module scandeval.datasets</i> ), 52
<code>load_absabank_imm()</code> ( <i>in module scandeval.datasets</i> ), 49	<code>load_ndt_nn_pos()</code> ( <i>in module scandeval.datasets</i> ), 52
<code>load_angry_tweets()</code> ( <i>in module scandeval.datasets</i> ), 50	<code>load_nordial()</code> ( <i>in module scandeval.datasets</i> ), 53
<code>load_dalaj()</code> ( <i>in module scandeval.datasets</i> ), 50	<code>load_norec()</code> ( <i>in module scandeval.datasets</i> ), 53
<code>load_dane()</code> ( <i>in module scandeval.datasets</i> ), 50	<code>load_norec_fo()</code> ( <i>in module scandeval.datasets</i> ), 53
<code>load_dataset()</code> ( <i>in module scandeval.datasets</i> ), 50	<code>load_norec_is()</code> ( <i>in module scandeval.datasets</i> ), 53
<code>load_ddt_dep()</code> ( <i>in module scandeval.datasets</i> ), 50	<code>load_norne_nb()</code> ( <i>in module scandeval.datasets</i> ), 53
<code>load_ddt_pos()</code> ( <i>in module scandeval.datasets</i> ), 50	<code>load_norne_nn()</code> ( <i>in module scandeval.datasets</i> ), 53
<code>load_dkhate()</code> ( <i>in module scandeval.datasets</i> ), 51	<code>load_sdt_dep()</code> ( <i>in module scandeval.datasets</i> ), 54
<code>load_europarl()</code> ( <i>in module scandeval.datasets</i> ), 51	<code>load_sdt_pos()</code> ( <i>in module scandeval.datasets</i> ), 54
<code>load_fdt_dep()</code> ( <i>in module scandeval.datasets</i> ), 51	<code>load_suc3()</code> ( <i>in module scandeval.datasets</i> ), 54
<code>load_fdt_pos()</code> ( <i>in module scandeval.datasets</i> ), 51	
<code>load_idt_dep()</code> ( <i>in module scandeval.datasets</i> ), 51	
<code>load_idt_pos()</code> ( <i>in module scandeval.datasets</i> ), 51	
<code>load_lcc()</code> ( <i>in module scandeval.datasets</i> ), 52	
<code>load_mim_gold_ner()</code> ( <i>in module scandeval.datasets</i> ), 52	
<code>load_ndt_nb_dep()</code> ( <i>in module scandeval.datasets</i> ), 52	
<code>load_ndt_nb_pos()</code> ( <i>in module scandeval.datasets</i> ), 52	
<code>load_ndt_nn_dep()</code> ( <i>in module scandeval.datasets</i> ), 52	
<code>load_ndt_nn_pos()</code> ( <i>in module scandeval.datasets</i> ), 52	
<code>load_nordial()</code> ( <i>in module scandeval.datasets</i> ), 53	
<code>load_norec()</code> ( <i>in module scandeval.datasets</i> ), 53	
<code>load_norec_fo()</code> ( <i>in module scandeval.datasets</i> ), 53	
<code>load_norec_is()</code> ( <i>in module scandeval.datasets</i> ), 53	
<code>load_norne_nb()</code> ( <i>in module scandeval.datasets</i> ), 53	
<code>load_norne_nn()</code> ( <i>in module scandeval.datasets</i> ), 53	
<code>load_sdt_dep()</code> ( <i>in module scandeval.datasets</i> ), 54	
<code>load_sdt_pos()</code> ( <i>in module scandeval.datasets</i> ), 54	
<code>load_suc3()</code> ( <i>in module scandeval.datasets</i> ), 54	

load_twitter_sent() (in module scandeval.datasets), 54	val.benchmarks.idt_pos.IdtPosBenchmark (attribute), 25
load_wikiann_fo() (in module scandeval.datasets), 54	metric_names (scandeval.benchmarks.lcc.LccBenchmark (attribute)), 26
<b>M</b>	
metric_names (scandeval.benchmarks.absabank_imm.AbsabankImmBenchmark (attribute)), 11	metric_names (scandeval.benchmarks.mim_gold_ner.MimGoldNerBenchmark (attribute)), 28
metric_names (scandeval.benchmarks.abstract.dep.DepBenchmark (attribute)), 3	metric_names (scandeval.benchmarks.ndt_nb_dep.NdtNBDepBenchmark (attribute)), 29
metric_names (scandeval.benchmarks.abstract.ner.NerBenchmark (attribute)), 4	metric_names (scandeval.benchmarks.ndt_nb_pos.NdtNBPosBenchmark (attribute)), 30
metric_names (scandeval.benchmarks.abstract.pos.PosBenchmark (attribute)), 6	metric_names (scandeval.benchmarks.ndt_nn_dep.NdtNNDepBenchmark (attribute)), 31
metric_names (scandeval.benchmarks.abstract.text_classification.TextClassificationBenchmark (attribute)), 8	metric_names (scandeval.benchmarks.ndt_nn_pos.NdtNNPosBenchmark (attribute)), 33
metric_names (scandeval.benchmarks.abstract.token_classification.TokenClassificationBenchmark (attribute)), 10	metric_names (scandeval.benchmarks.nordial.NorDialBenchmark (attribute)), 34
metric_names (scandeval.benchmarks.angry_tweets.AngryTweetsBenchmark (attribute)), 13	metric_names (scandeval.benchmarks.norec.NorecBenchmark (attribute)), 35
metric_names (scandeval.benchmarks.dalaj.DalajBenchmark (attribute)), 14	metric_names (scandeval.benchmarks.norec_fo.NorecFOBenchmark (attribute)), 36
metric_names (scandeval.benchmarks.dane.DaneBenchmark (attribute)), 15	metric_names (scandeval.benchmarks.norec_is.NorecISBenchmark (attribute)), 38
metric_names (scandeval.benchmarks.ddt_dep.DdtDepBenchmark (attribute)), 16	metric_names (scandeval.benchmarks.norne_nb.NorneNBBenchmark (attribute)), 39
metric_names (scandeval.benchmarks.ddt_pos.DdtPosBenchmark (attribute)), 18	metric_names (scandeval.benchmarks.norne_nn.NorneNNBenchmark (attribute)), 40
metric_names (scandeval.benchmarks.dkhate.DkHateBenchmark (attribute)), 19	metric_names (scandeval.benchmarks.sdt_dep.SdtDepBenchmark (attribute)), 41
metric_names (scandeval.benchmarks.europarl.EuroparlBenchmark (attribute)), 20	metric_names (scandeval.benchmarks.sdt_pos.SdtPosBenchmark (attribute)), 43
metric_names (scandeval.benchmarks.fdt_dep.FdtDepBenchmark (attribute)), 21	metric_names (scandeval.benchmarks.suc3.Suc3Benchmark (attribute)), 44
metric_names (scandeval.benchmarks.fdt_pos.FdtPosBenchmark (attribute)), 23	metric_names (scandeval.benchmarks.twitter_sent.TwitterSentBenchmark (attribute)), 45
metric_names (scandeval.benchmarks.idt_dep.IdtDepBenchmark (attribute)), 24	metric_names (scandeval.benchmarks.wikiann_fo.WikiannFoBenchmark (attribute)), 47
metric_names (scandeval.benchmarks.mim_gold_ner.MimGoldNerBenchmark (class in scandeval.benchmarks)), 28	

*val.benchmarks.mim\_gold\_ner*), 27

module

- scandeval, 56
- scandeval.benchmark, 47
- scandeval.benchmarks, 47
- scandeval.benchmarks.absabank\_imm, 11
- scandeval.benchmarks.abstract, 11
- scandeval.benchmarks.abstract.base, 1
- scandeval.benchmarks.abstract.dep, 3
- scandeval.benchmarks.abstract.ner, 4
- scandeval.benchmarks.abstract.pos, 5
- scandeval.benchmarks.abstract.text\_classification, 7
- scandeval.benchmarks.abstract.token\_classification, 9
- scandeval.benchmarks.angry\_tweets, 12
- scandeval.benchmarks.dalaj, 14
- scandeval.benchmarks.dane, 15
- scandeval.benchmarks.ddt\_dep, 16
- scandeval.benchmarks.ddt\_pos, 17
- scandeval.benchmarks.dkhate, 19
- scandeval.benchmarks.europarl, 20
- scandeval.benchmarks.fdt\_dep, 21
- scandeval.benchmarks.fdt\_pos, 22
- scandeval.benchmarks.idt\_dep, 24
- scandeval.benchmarks.idt\_pos, 25
- scandeval.benchmarks.lcc, 26
- scandeval.benchmarks.mim\_gold\_ner, 27
- scandeval.benchmarks.ndt\_nb\_dep, 29
- scandeval.benchmarks.ndt\_nb\_pos, 30
- scandeval.benchmarks.ndt\_nn\_dep, 31
- scandeval.benchmarks.ndt\_nn\_pos, 32
- scandeval.benchmarks.nordial, 34
- scandeval.benchmarks.norec, 35
- scandeval.benchmarks.norec\_fo, 36
- scandeval.benchmarks.norec\_is, 37
- scandeval.benchmarks.norne\_nb, 39
- scandeval.benchmarks.norne\_nn, 40
- scandeval.benchmarks.sdt\_dep, 41
- scandeval.benchmarks.sdt\_pos, 42
- scandeval.benchmarks.suc3, 44
- scandeval.benchmarks.twitter\_sent, 45
- scandeval.benchmarks.wikiann\_fo, 46
- scandeval.cli, 49
- scandeval.datasets, 49
- scandeval.utils, 55

## N

name (*scandeval.benchmarks.absabank\_imm.AbsabankImmBenchmark* attribute), 11

name (*scandeval.benchmarks.abstract.dep.DepBenchmark* attribute), 3

name (*scandeval.benchmarks.abstract.ner.NerBenchmark* attribute), 4

name (*scandeval.benchmarks.abstract.pos.PosBenchmark* attribute), 5

name (*scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark* attribute), 8

name (*scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark* attribute), 10

name (*scandeval.benchmarks.angry\_tweets.AngryTweetsBenchmark* attribute), 13

name (*scandeval.benchmarks.dalaj.DalajBenchmark* attribute), 14

name (*scandeval.benchmarks.dane.DaneBenchmark* attribute), 15

name (*scandeval.benchmarks.ddt\_dep.DdtDepBenchmark* attribute), 16

name (*scandeval.benchmarks.ddt\_pos.DdtPosBenchmark* attribute), 18

name (*scandeval.benchmarks.dkhate.DkHateBenchmark* attribute), 19

name (*scandeval.benchmarks.europarl.EuroparlBenchmark* attribute), 20

name (*scandeval.benchmarks.fdt\_dep.FdtDepBenchmark* attribute), 21

name (*scandeval.benchmarks.fdt\_pos.FdtPosBenchmark* attribute), 23

name (*scandeval.benchmarks.idt\_dep.IdtDepBenchmark* attribute), 24

name (*scandeval.benchmarks.idt\_pos.IdtPosBenchmark* attribute), 25

name (*scandeval.benchmarks.lcc.LccBenchmark* attribute), 26

name (*scandeval.benchmarks.mim\_gold\_ner.MimGoldNerBenchmark* attribute), 28

name (*scandeval.benchmarks.ndt\_nb\_dep.NdtNBDepBenchmark* attribute), 29

name (*scandeval.benchmarks.ndt\_nb\_pos.NdtNBPosBenchmark* attribute), 30

name (*scandeval.benchmarks.ndt\_nn\_dep.NdtNNDepBenchmark* attribute), 31

name (*scandeval.benchmarks.ndt\_nn\_pos.NdtNNPosBenchmark* attribute), 33

name (*scandeval.benchmarks.nordial.NorDialBenchmark* attribute), 34

name (*scandeval.benchmarks.norec.NorecBenchmark* attribute), 35

name (*scandeval.benchmarks.norec\_fo.NorecFOBenchmark* attribute), 36

name (*scandeval.benchmarks.norec\_is.NorecISBenchmark* attribute), 38

name (*scandeval.benchmarks.norne\_nb.NorneNBBenchmark* attribute), 39

name (*scandeval.benchmarks.norne\_nn.NorneNNBenchmark* attribute), 40

name (*scandeval.benchmarks.sdt\_dep.SdtDepBenchmark* attribute), 41

```

name (scandeval.benchmarks.sdt_pos.SdtPosBenchmark
      attribute), 43
name (scandeval.benchmarks.suc3.Suc3Benchmark
      attribute), 44
name (scandeval.benchmarks.twitter_sent.TwitterSentBenchmark
      attribute), 45
name (scandeval.benchmarks.wikiann_fo.WikiannFoBenchmark
      attribute), 46
NdtNBDepBenchmark (class in
                    val.benchmarks.ndt_nb_dep), 29
NdtNBPosBenchmark (class in
                    val.benchmarks.ndt_nb_pos), 30
NdtNNDepBenchmark (class in
                    val.benchmarks.ndt_nn_dep), 31
NdtNNPosBenchmark (class in
                    val.benchmarks.ndt_nn_pos), 32
NerBenchmark (class in
               val.benchmarks.abstract.ner), 4
NeverLeaveProgressCallback (class in
                           val.utils), 55
NorDialBenchmark (class in
                  val.benchmarks.nordial), 34
NorecBenchmark (class in
                 val.benchmarks.norec), 35
NorecFOBenchmark (class in
                  val.benchmarks.norec_fo), 36
NorecISBenchmark (class in
                  val.benchmarks.norec_is), 37
NorneNBBenchmark (class in
                  val.benchmarks.norne_nb), 39
NorneNNBenchmark (class in
                   val.benchmarks.norne_nn), 40
num_labels (scandeval.benchmarks.absabank_imm.AbsabankImmBenchmark
             attribute), 12
num_labels (scandeval.benchmarks.abstract.dep.DepBenchmark
             attribute), 3
num_labels (scandeval.benchmarks.abstract.ner.NerBenchmark
             attribute), 5
num_labels (scandeval.benchmarks.abstract.pos.PosBenchmark
             attribute), 6
num_labels (scandeval.benchmarks.abstract.text_classification.TextClassificationBenchmark
             attribute), 8
num_labels (scandeval.benchmarks.abstract.token_classification.TokenClassificationBenchmark
             attribute), 10
num_labels (scandeval.benchmarks.angry_tweets.AngryTweetsBenchmark
             attribute), 13
num_labels (scandeval.benchmarks.dalaj.DalajBenchmark
             attribute), 14
num_labels (scandeval.benchmarks.dane.DaneBenchmark
             attribute), 15
num_labels (scandeval.benchmarks.ddt_dep.DdtDepBenchmark
             attribute), 17
num_labels (scandeval.benchmarks.ddt_pos.DdtPosBenchmark
             attribute), 18
num_labels (scandeval.benchmarks.dkhate.DkHateBenchmark
             attribute), 19
num_labels (scandeval.benchmarks.europarl.EuroparlBenchmark
             attribute), 20
num_labels (scandeval.benchmarks.fdt_dep.FdtDepBenchmark
             attribute), 22
num_labels (scandeval.benchmarks.fdt_pos.FdtPosBenchmark
             attribute), 23
scandeval.benchmarks.idt_dep.IdtDepBenchmark
            (class in val.benchmarks.idt_dep), 24
scandeval.benchmarks.idt_pos.IdtPosBenchmark
            (class in val.benchmarks.idt_pos), 25
scandeval.benchmarks.lcc.LccBenchmark
            (class in val.benchmarks.lcc), 27
scandeval.benchmarks.mim_gold_ner.MimGoldNerBenchmark
            (class in val.benchmarks.mim_gold_ner), 28
scandeval.benchmarks.ndt_nb_dep.NdtNBDepBenchmark
            (class in val.benchmarks.ndt_nb_dep), 29
scandeval.benchmarks.ndt_nb_pos.NdtNBPosBenchmark
            (class in val.benchmarks.ndt_nb_pos), 30
scandeval.benchmarks.ndt_nn_dep.NdtNNDepBenchmark
            (class in val.benchmarks.ndt_nn_dep), 32
scandeval.benchmarks.ndt_nn_pos.NdtNNPosBenchmark
            (class in val.benchmarks.ndt_nn_pos), 33
scandeval.benchmarks.nordial.NorDialBenchmark
            (class in val.benchmarks.nordial), 34
scandeval.benchmarks.norec.NorecBenchmark
            (class in val.benchmarks.norec), 35
scandeval.benchmarks.norec_fo.NorecFOBenchmark
            (class in val.benchmarks.norec_fo), 36
scandeval.benchmarks.norec_is.NorecISBenchmark
            (class in val.benchmarks.norec_is), 37
scandeval.benchmarks.norne_nb.NorneNBBenchmark
            (class in val.benchmarks.norne_nb), 39
scandeval.benchmarks.norne_nn.NorneNNBenchmark
            (class in val.benchmarks.norne_nn), 40
num_labels (scandeval.benchmarks.norne_nb.NorneNBBenchmark
             attribute), 39
num_labels (scandeval.benchmarks.norne_nn.NorneNNBenchmark
             attribute), 40
num_labels (scandeval.benchmarks.sdt_dep.SdtDepBenchmark
             attribute), 42
num_labels (scandeval.benchmarks.sdt_pos.SdtPosBenchmark
             attribute), 43
num_labels (scandeval.benchmarks.suc3.Suc3Benchmark
             attribute), 44
num_labels (scandeval.benchmarks.twitter_sent.TwitterSentBenchmark
             attribute), 45
num_labels (scandeval.benchmarks.wikiann_fo.WikiannFoBenchmark
             attribute), 47
on_prediction_step() (scandeval.utils.NeverLeaveProgressCallback
                     method), 55
on_train_begin() (scandeval.utils.NeverLeaveProgressCallback
                  method), 55

```



## P

PosBenchmark (class in scandeval.benchmarks.abstract.pos), 5  
 progress\_bar (scandeval.benchmark.Benchmark attribute), 48

## S

save\_results (scandeval.benchmark.Benchmark attribute), 48  
 scandeval  
   module, 56  
 scandeval.benchmark  
   module, 47  
 scandeval.benchmarks  
   module, 47  
 scandeval.benchmarks.absabank\_imm  
   module, 11  
 scandeval.benchmarks.abstract  
   module, 11  
 scandeval.benchmarks.abstract.base  
   module, 1  
 scandeval.benchmarks.abstract.dep  
   module, 3  
 scandeval.benchmarks.abstract.ner  
   module, 4  
 scandeval.benchmarks.abstract.pos  
   module, 5  
 scandeval.benchmarks.abstract.text\_classification  
   module, 7  
 scandeval.benchmarks.abstract.token\_classification  
   module, 9  
 scandeval.benchmarks.angry\_tweets  
   module, 12  
 scandeval.benchmarks.dalaj  
   module, 14  
 scandeval.benchmarks.dane  
   module, 15  
 scandeval.benchmarks.ddt\_dep  
   module, 16  
 scandeval.benchmarks.ddt\_pos  
   module, 17  
 scandeval.benchmarks.dkhate  
   module, 19  
 scandeval.benchmarks.europarl  
   module, 20  
 scandeval.benchmarks.fdt\_dep  
   module, 21  
 scandeval.benchmarks.fdt\_pos  
   module, 22  
 scandeval.benchmarks.idt\_dep  
   module, 24  
 scandeval.benchmarks.idt\_pos  
   module, 25  
 scandeval.benchmarks.lcc  
   module, 26  
 scandeval.benchmarks.mim\_gold\_ner  
   module, 27  
 scandeval.benchmarks.ndt\_nb\_dep  
   module, 29  
 scandeval.benchmarks.ndt\_nb\_pos  
   module, 30  
 scandeval.benchmarks.ndt\_nn\_dep  
   module, 31  
 scandeval.benchmarks.ndt\_nn\_pos  
   module, 32  
 scandeval.benchmarks.nordial  
   module, 34  
 scandeval.benchmarks.norec  
   module, 35  
 scandeval.benchmarks.norec\_fo  
   module, 36  
 scandeval.benchmarks.norec\_is  
   module, 37  
 scandeval.benchmarks.norne\_nb  
   module, 39  
 scandeval.benchmarks.norne\_nn  
   module, 40  
 scandeval.benchmarks.sdt\_dep  
   module, 41  
 scandeval.benchmarks.sdt\_pos  
   module, 42  
 scandeval.benchmarks.suc3  
   module, 44  
 scandeval.benchmarks.twitter\_sent  
   module, 45  
 scandeval.benchmarks.wikiann\_fo  
   module, 46  
 scandeval.cli  
   module, 49  
 scandeval.datasets  
   module, 49  
 scandeval.utils  
   module, 55  
 SdtDepBenchmark (class in scandeval.benchmarks.sdt\_dep), 41  
 SdtPosBenchmark (class in scandeval.benchmarks.sdt\_pos), 42  
 split\_point (scandeval.benchmarks.absabank\_imm.AbsabankImmBenchmark attribute), 12  
 split\_point (scandeval.benchmarks.abstract.dep.DepBenchmark attribute), 4  
 split\_point (scandeval.benchmarks.abstract.ner.NerBenchmark attribute), 5  
 split\_point (scandeval.benchmarks.abstract.pos.PosBenchmark

- attribute*), 6  
 split\_point (scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark attribute), 8  
 split\_point (scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark attribute), 11  
 split\_point (scandeval.benchmarks.angry\_tweets.AngryTweetsBenchmark attribute), 13  
 split\_point (scandeval.benchmarks.dalaj.DalajBenchmark attribute), 15  
 split\_point (scandeval.benchmarks.dane.DaneBenchmark attribute), 16  
 split\_point (scandeval.benchmarks.ddt\_dep.DdtDepBenchmark attribute), 17  
 split\_point (scandeval.benchmarks.ddt\_pos.DdtPosBenchmark attribute), 18  
 split\_point (scandeval.benchmarks.dkhate.DkHateBenchmark attribute), 20  
 split\_point (scandeval.benchmarks.europarl.EuroparlBenchmark attribute), 21  
 split\_point (scandeval.benchmarks.fdt\_dep.FdtDepBenchmark attribute), 22  
 split\_point (scandeval.benchmarks.fdt\_pos.FdtPosBenchmark attribute), 23  
 split\_point (scandeval.benchmarks.idt\_dep.IdtDepBenchmark attribute), 25  
 split\_point (scandeval.benchmarks.idt\_pos.IdtPosBenchmark attribute), 26  
 split\_point (scandeval.benchmarks.lcc.LccBenchmark attribute), 27  
 split\_point (scandeval.benchmarks.mim\_gold\_ner.MimGoldNerBenchmark attribute), 28  
 split\_point (scandeval.benchmarks.ndt\_nb\_dep.NdtNBDepBenchmark attribute), 30  
 split\_point (scandeval.benchmarks.ndt\_nb\_pos.NdtNBPosBenchmark attribute), 31  
 split\_point (scandeval.benchmarks.ndt\_nn\_dep.NdtNNDepBenchmark attribute), 32  
 split\_point (scandeval.benchmarks.ndt\_nn\_pos.NdtNNPosBenchmark attribute), 33  
 split\_point (scandeval.benchmarks.nordial.NorDialBenchmark attribute), 35  
 split\_point (scandeval.benchmarks.norec.NorecBenchmark attribute), 36  
 split\_point (scandeval.benchmarks.norec\_fo.NorecFOBenchmark attribute), 37  
 split\_point (scandeval.benchmarks.norec\_is.NorecISBenchmark attribute), 38  
 split\_point (scandeval.benchmarks.norne\_nb.NorneNBBenchmark attribute), 40  
 split\_point (scandeval.benchmarks.norne\_nn.NorneNNBenchmark attribute), 41  
 split\_point (scandeval.benchmarks.sdt\_dep.SdtDepBenchmark attribute), 42  
 split\_point (scandeval.benchmarks.sdt\_pos.SdtPosBenchmark attribute), 43  
 split\_point (scandeval.benchmarks.suc3.Suc3Benchmark attribute), 45  
 split\_point (scandeval.benchmarks.twitter\_sent.TwitterSentBenchmark attribute), 46  
 split\_point (scandeval.benchmarks.wikiann\_fo.WikiannFoBenchmark attribute), 47  
 Suc3Benchmark (class in scandeval.benchmarks.suc3), 44
- ## T
- task (scandeval.benchmark.Benchmark attribute), 48  
 task (scandeval.benchmarks.absabank\_imm.AbsabankImmBenchmark attribute), 11  
 task (scandeval.benchmarks.abstract.dep.DepBenchmark attribute), 3  
 task (scandeval.benchmarks.abstract.ner.NerBenchmark attribute), 4  
 task (scandeval.benchmarks.abstract.pos.PosBenchmark attribute), 6  
 task (scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark attribute), 8  
 task (scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark attribute), 10

task (*scandeval.benchmarks.angry\_tweets.AngryTweetsBenchmark* attribute), 13

task (*scandeval.benchmarks.dalaj.DalajBenchmark* attribute), 14

task (*scandeval.benchmarks.dane.DaneBenchmark* attribute), 15

task (*scandeval.benchmarks.ddt\_dep.DdtDepBenchmark* attribute), 16

task (*scandeval.benchmarks.ddt\_pos.DdtPosBenchmark* attribute), 18

task (*scandeval.benchmarks.dkhate.DkHateBenchmark* attribute), 19

task (*scandeval.benchmarks.europarl.EuroparlBenchmark* attribute), 20

task (*scandeval.benchmarks.fdt\_dep.FdtDepBenchmark* attribute), 21

task (*scandeval.benchmarks.fdt\_pos.FdtPosBenchmark* attribute), 23

task (*scandeval.benchmarks.idt\_dep.IdtDepBenchmark* attribute), 24

task (*scandeval.benchmarks.idt\_pos.IdtPosBenchmark* attribute), 25

task (*scandeval.benchmarks.lcc.LccBenchmark* attribute), 26

task (*scandeval.benchmarks.mim\_gold\_ner.MimGoldNerBenchmark* attribute), 28

task (*scandeval.benchmarks.ndt\_nb\_dep.NdtNBDepBenchmark* attribute), 29

task (*scandeval.benchmarks.ndt\_nb\_pos.NdtNBPosBenchmark* attribute), 30

task (*scandeval.benchmarks.ndt\_nn\_dep.NdtNNDepBenchmark* attribute), 31

task (*scandeval.benchmarks.ndt\_nn\_pos.NdtNNPosBenchmark* attribute), 33

task (*scandeval.benchmarks.nordial.NorDialBenchmark* attribute), 34

task (*scandeval.benchmarks.norec.NorecBenchmark* attribute), 35

task (*scandeval.benchmarks.norec\_fo.NorecFOBenchmark* attribute), 36

task (*scandeval.benchmarks.norec\_is.NorecISBenchmark* attribute), 38

task (*scandeval.benchmarks.norne\_nb.NorneNBBenchmark* attribute), 39

task (*scandeval.benchmarks.norne\_nn.NorneNNBenchmark* attribute), 40

task (*scandeval.benchmarks.sdt\_dep.SdtDepBenchmark* attribute), 41

task (*scandeval.benchmarks.sdt\_pos.SdtPosBenchmark* attribute), 43

task (*scandeval.benchmarks.suc3.Suc3Benchmark* attribute), 44

task (*scandeval.benchmarks.twitter\_sent.TwitterSentBenchmark* attribute), 45

task (*scandeval.benchmarks.wikiann\_fo.WikiannFoBenchmark* attribute), 46

TextClassificationBenchmark (class in *scandeval.benchmarks.abstract.text\_classification*), 7

TokenClassificationBenchmark (class in *scandeval.benchmarks.abstract.token\_classification*), 9

TwitterSentBenchmark (class in *scandeval.benchmarks.twitter\_sent*), 45

two\_labels (*scandeval.benchmarks.absabank\_imm.AbsabankImmBenchmark* attribute), 12

two\_labels (*scandeval.benchmarks.abstract.dep.DepBenchmark* attribute), 4

two\_labels (*scandeval.benchmarks.abstract.ner.NerBenchmark* attribute), 5

two\_labels (*scandeval.benchmarks.abstract.pos.PosBenchmark* attribute), 6

two\_labels (*scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark* attribute), 8

two\_labels (*scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark* attribute), 11

two\_labels (*scandeval.benchmarks.angry\_tweets.AngryTweetsBenchmark* attribute), 13

two\_labels (*scandeval.benchmarks.dalaj.DalajBenchmark* attribute), 15

two\_labels (*scandeval.benchmarks.dane.DaneBenchmark* attribute), 16

two\_labels (*scandeval.benchmarks.ddt\_dep.DdtDepBenchmark* attribute), 17

two\_labels (*scandeval.benchmarks.ddt\_pos.DdtPosBenchmark* attribute), 18

two\_labels (*scandeval.benchmarks.dkhate.DkHateBenchmark* attribute), 19

two\_labels (*scandeval.benchmarks.europarl.EuroparlBenchmark* attribute), 21

two\_labels (*scandeval.benchmarks.fdt\_dep.FdtDepBenchmark* attribute), 22

two\_labels (*scandeval.benchmarks.fdt\_pos.FdtPosBenchmark* attribute), 23

two\_labels (*scandeval.benchmarks.idt\_dep.IdtDepBenchmark* attribute), 24

two\_labels (*scandeval.benchmarks.idt\_pos.IdtPosBenchmark* attribute), 26

two\_labels (*scandeval.benchmarks.lcc.LccBenchmark* attribute), 27

two\_labels (*scandeval.benchmarks.mim\_gold\_ner.MimGoldNerBenchmark* attribute), 28

two\_labels (*scandeval.benchmarks.ndt\_nb\_dep.NdtNBDepBenchmark* attribute), 29

two\_labels (*scandeval.benchmarks.ndt\_nb\_pos.NdtNBPosBenchmark* attribute), 31

two\_labels (*scandeval.benchmarks.ndt\_nn\_dep.NdtNNDepBenchmark* attribute), 32

two\_labels (scandeval.benchmarks.ndt\_nn\_pos.NdtNNPosBenchmark attribute), 33

two\_labels (scandeval.benchmarks.nordial.NorDialBenchmark attribute), 34

two\_labels (scandeval.benchmarks.norec.NorecBenchmark attribute), 36

two\_labels (scandeval.benchmarks.norec\_fo.NorecFOBenchmark attribute), 37

two\_labels (scandeval.benchmarks.norec\_is.NorecISBenchmark attribute), 38

two\_labels (scandeval.benchmarks.norne\_nb.NorneNBBenchmark attribute), 39

two\_labels (scandeval.benchmarks.norne\_nn.NorneNNBenchmark attribute), 41

two\_labels (scandeval.benchmarks.sdt\_dep.SdtDepBenchmark attribute), 42

two\_labels (scandeval.benchmarks.sdt\_pos.SdtPosBenchmark attribute), 43

two\_labels (scandeval.benchmarks.suc3.Suc3Benchmark attribute), 44

two\_labels (scandeval.benchmarks.twitter\_sent.TwitterSentBenchmark attribute), 46

two\_labels (scandeval.benchmarks.wikiann\_fo.WikiannFoBenchmark attribute), 47

TwolabelTrainer (class in scandeval.utils), 55

**U**

use\_parent\_doc() (scandeval.utils.DocInherit method), 55

**V**

verbose (scandeval.benchmark.Benchmark attribute), 48

verbose (scandeval.benchmarks.absabank\_imm.AbsabankImmBenchmark attribute), 12

verbose (scandeval.benchmarks.abstract.dep.DepBenchmark attribute), 4

verbose (scandeval.benchmarks.abstract.ner.NerBenchmark attribute), 5

verbose (scandeval.benchmarks.abstract.pos.PosBenchmark attribute), 6

verbose (scandeval.benchmarks.abstract.text\_classification.TextClassificationBenchmark attribute), 8

verbose (scandeval.benchmarks.abstract.token\_classification.TokenClassificationBenchmark attribute), 11

verbose (scandeval.benchmarks.angry\_tweets.AngryTweetsBenchmark attribute), 13

verbose (scandeval.benchmarks.dalaj.DalajBenchmark attribute), 15

verbose (scandeval.benchmarks.dane.DaneBenchmark attribute), 16

verbose (scandeval.benchmarks.ddt\_dep.DdtDepBenchmark attribute), 17

verbose (scandeval.benchmarks.ddt\_pos.DdtPosBenchmark attribute), 18

verbose (scandeval.benchmarks.dkhate.DkHateBenchmark attribute), 20

verbose (scandeval.benchmarks.europarl.EuroparlBenchmark attribute), 21

verbose (scandeval.benchmarks.fdt\_dep.FdtDepBenchmark attribute), 22

verbose (scandeval.benchmarks.fdt\_pos.FdtPosBenchmark attribute), 23

verbose (scandeval.benchmarks.idt\_dep.IdtDepBenchmark attribute), 25

verbose (scandeval.benchmarks.idt\_pos.IdtPosBenchmark attribute), 26

verbose (scandeval.benchmarks.lcc.LccBenchmark attribute), 27

verbose (scandeval.benchmarks.mim\_gold\_ner.MimGoldNerBenchmark attribute), 28

verbose (scandeval.benchmarks.ndt\_nb\_dep.NdtNBDepBenchmark attribute), 30

verbose (scandeval.benchmarks.ndt\_nb\_pos.NdtNBPosBenchmark attribute), 31

verbose (scandeval.benchmarks.ndt\_nn\_dep.NdtNNDepBenchmark attribute), 32

verbose (scandeval.benchmarks.ndt\_nn\_pos.NdtNNPosBenchmark attribute), 33

verbose (scandeval.benchmarks.nordial.NorDialBenchmark attribute), 35

verbose (scandeval.benchmarks.norec.NorecBenchmark attribute), 36

verbose (scandeval.benchmarks.norec\_fo.NorecFOBenchmark attribute), 37

verbose (scandeval.benchmarks.norec\_is.NorecISBenchmark attribute), 38

verbose (scandeval.benchmarks.norne\_nb.NorneNBBenchmark attribute), 40

verbose (scandeval.benchmarks.norne\_nn.NorneNNBenchmark attribute), 41

verbose (scandeval.benchmarks.sdt\_dep.SdtDepBenchmark attribute), 42

verbose (scandeval.benchmarks.sdt\_pos.SdtPosBenchmark attribute), 43

verbose (scandeval.benchmarks.suc3.Suc3Benchmark attribute), 45

verbose (scandeval.benchmarks.twitter\_sent.TwitterSentBenchmark attribute), 46

WikiannFoBenchmark (class in scandeval.benchmarks.wikiann\_fo), 46