

---

# MatchZoo Documentation

*Release 1.0*

**MatchZoo**

**Aug 20, 2019**



---

## Contents:

---

<b>1</b>	<b>matchzoo</b>	<b>3</b>
<b>2</b>	<b>MatchZoo Model Reference</b>	<b>5</b>
2.1	DenseBaseline . . . . .	5
2.2	DSSM . . . . .	6
2.3	CDSSM . . . . .	7
2.4	DRMM . . . . .	8
2.5	DRMMTKS . . . . .	9
2.6	ESIM . . . . .	10
2.7	KNRM . . . . .	11
2.8	ConvKNRM . . . . .	12
2.9	BiMPM . . . . .	13
2.10	MatchLSTM . . . . .	14
2.11	ArcII . . . . .	15
<b>3</b>	<b>API Reference</b>	<b>17</b>
3.1	matchzoo . . . . .	17
<b>4</b>	<b>Indices and tables</b>	<b>169</b>
	<b>Python Module Index</b>	<b>171</b>
	<b>Index</b>	<b>173</b>





MatchZoo is a toolkit for text matching. It was developed with a focus on facilitating the designing, comparing and sharing of deep text matching models. There are a number of deep matching methods, such as DRMM, MatchPyramid, MV-LSTM, aNMM, DUET, ARC-I, ARC-II, DSSM, and CDSSM, designed with a unified interface. Potential tasks related to MatchZoo include document retrieval, question answering, conversational response ranking, paraphrase identification, etc. We are always happy to receive any code contributions, suggestions, comments from all our MatchZoo users.



# CHAPTER 1

---

matchzoo

---





## 2.1 DenseBaseline

### 2.1.1 Model Documentation

A simple densely connected baseline model.

**Examples:**

```
>>> model = DenseBaseline()
>>> model.params['mlp_num_layers'] = 2
>>> model.params['mlp_num_units'] = 300
>>> model.params['mlp_num_fan_out'] = 128
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.1.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.dense_baseline.DenseBaseline'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_embedding	A flag used help <i>auto</i> module. Shouldn't be changed.	True	
3	embedding	FloatTensor containing weights for the Embedding.		
4	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
5	embedding_output_dim	Should be set manually.		
6	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
7	with_multi_layer_perceptron	A flag option whether a multiple layer perceptron is used. Shouldn't be changed.	True	
8	mlp_num_units	Number of units in first <i>mlp_num_layers</i> layers.	256	quantitative uniform distribution in [16, 512), with a step size of 1
9	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 5), with a step size of 1
10	mlp_num_fan_out	Number of units of the layer that connects the multiple layer perceptron and the output.	64	quantitative uniform distribution in [4, 128), with a step size of 4
11	mlp_activation_func	Activation function used in the multiple layer perceptron.	relu	

## 2.2 DSSM

### 2.2.1 Model Documentation

Deep structured semantic model.

**Examples:**

```
>>> model = DSSM()
>>> model.params['mlp_num_layers'] = 3
>>> model.params['mlp_num_units'] = 300
>>> model.params['mlp_num_fan_out'] = 128
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.2.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.dssm.DSSM'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_multi_layer_perceptron	A flag to indicate whether a multiple layer perceptron is used. Shouldn't be changed.	True	
3	mlp_num_units	Number of units in first <i>mlp_num_layers</i> layers.	128	quantitative uniform distribution in [8, 256), with a step size of 8
4	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
5	mlp_num_fan_out	Number of units of the layer that connects the multiple layer perceptron and the output.	64	quantitative uniform distribution in [4, 128), with a step size of 4
6	mlp_activation_func	Activation function used in the multiple layer perceptron.	relu	
7	vocab_size	Size of vocabulary.	379	

## 2.3 CDSSM

### 2.3.1 Model Documentation

CDSSM Model implementation.

Learning Semantic Representations Using Convolutional Neural Networks for Web Search. (2014a) A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. (2014b)

**Examples:**

```
>>> import matchzoo as mz
>>> model = CDSSM()
>>> model.params['task'] = mz.tasks.Ranking()
>>> model.params['vocab_size'] = 4
>>> model.params['filters'] = 32
>>> model.params['kernel_size'] = 3
>>> model.params['conv_activation_func'] = 'relu'
>>> model.build()
```

## 2.3.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.cdssm.CDSSM'>	CDSSM
1	task	Decides model output shape, loss, and metrics.		
2	with_multi_layer_perceptron	Whether a multiple layer perceptron is used. Shouldn't be changed.	True	
3	mlp_num_units	Number of units in first <i>mlp_num_layers</i> layers.	128	quantitative uniform distribution in [8, 256), with a step size of 8
4	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
5	mlp_num_fan_out	Number of units of the layer that connects the multiple layer perceptron and the output.	64	quantitative uniform distribution in [4, 128), with a step size of 4
6	mlp_activation_func	Activation function used in the multiple layer perceptron.	relu	
7	vocab_size	Size of vocabulary.	379	
8	filters	Number of filters in the 1D convolution layer.	3	
9	kernel_size	Number of kernel size in the 1D convolution layer.	3	
10	strides	Strides in the 1D convolution layer.	1	
11	padding	The padding mode in the convolution layer. It should be one of <i>same</i> , <i>valid</i> , and <i>causal</i> .	0	
12	conv_activation_func	Activation function in the convolution layer.	relu	
13	dropout_rate	The dropout rate.	0.3	

## 2.4 DRMM

### 2.4.1 Model Documentation

DRMM Model.

Examples:

```
>>> model = DRMM()
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 5
>>> model.params['mlp_num_fan_out'] = 1
>>> model.params['mlp_activation_func'] = 'tanh'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.4.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.drmm.DRMM'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_embedding	A flag used help <i>auto</i> module. Shouldn't be changed.	True	
3	embedding	FloatTensor containing weights for the Embedding.		
4	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
5	embedding_output_dim	Should be set manually.		
6	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
7	with_multi_layer_perceptron	A flag to determine whether a multiple layer perceptron is used. Shouldn't be changed.	True	
8	mlp_num_units	Number of units in first <i>mlp_num_layers</i> layers.	128	quantitative uniform distribution in [8, 256), with a step size of 8
9	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
10	mlp_num_fan_out	Number of units of the layer that connects the multiple layer perceptron and the output.	1	quantitative uniform distribution in [4, 128), with a step size of 4
11	mlp_activation_func	Activation function used in the multiple layer perceptron.	relu	
12	mask_value	The value to be masked from inputs.	0	
13	hist_bin_size	The number of bin size of the histogram.	30	

## 2.5 DRMMTKS

### 2.5.1 Model Documentation

DRMMTKS Model.

Examples:

```
>>> model = DRMMTKS()
>>> model.params['top_k'] = 10
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 5
>>> model.params['mlp_num_fan_out'] = 1
>>> model.params['mlp_activation_func'] = 'tanh'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.5.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.drmmmts.DRMMTKS'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_embedding	A flag used help <i>auto</i> module. Shouldn't be changed.	True	
3	embedding	FloatTensor containing weights for the Embedding.		
4	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
5	embedding_output_dim	Should be set manually.		
6	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
7	with_multi_layer_perceptron	A flag to specify whether a multiple layer perceptron is used. Shouldn't be changed.	True	
8	mlp_num_units	Number of units in first <i>mlp_num_layers</i> layers.	128	quantitative uniform distribution in [8, 256), with a step size of 8
9	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
10	mlp_num_fan_in	Number of units of the layer that connects the multiple layer perceptron and the output.	1	quantitative uniform distribution in [4, 128), with a step size of 4
11	mlp_activation	Activation function used in the multiple layer perceptron.	relu	
12	mask_value	The value to be masked from inputs.	0	
13	top_k	Size of top-k pooling layer.	10	quantitative uniform distribution in [2, 100), with a step size of 1

## 2.6 ESIM

### 2.6.1 Model Documentation

ESIM Model.

**Examples:**

```
>>> model = ESIM()
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.6.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.esim.ESIM'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_embedding	A flag used help <i>auto</i> module. Shouldn't be changed.	True	
3	embedding	FloatTensor containing weights for the Embedding.		
4	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
5	embedding_output_dim	Should be set manually.		
6	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
7	mask_value	The value to be masked from inputs.	0	
8	dropout	Dropout rate.	0.2	
9	hidden_size	Hidden size.	200	
10	lstm_layer	Number of LSTM layers	1	
11	drop_lstm	Whether dropout LSTM.	False	
12	concat_lstm	Whether concat intermediate outputs.	True	
13	rnn_type	Choose rnn type, lstm or gru.	lstm	

## 2.7 KNRM

### 2.7.1 Model Documentation

KNRM Model.

**Examples:**

```
>>> model = KNRM()
>>> model.params['kernel_num'] = 11
>>> model.params['sigma'] = 0.1
>>> model.params['exact_sigma'] = 0.001
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.7.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.knrm.KNRM'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_embedding	A flag used help <i>auto</i> module. Shouldn't be changed.	True	
3	embedding	FloatTensor containing weights for the Embedding.		
4	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
5	embedding_output_dim	Should be set manually.		
6	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
7	kernel_num	The number of RBF kernels.	11	quantitative uniform distribution in [5, 20), with a step size of 1
8	sigma	The <i>sigma</i> defines the kernel width.	0.1	quantitative uniform distribution in [0.01, 0.2), with a step size of 0.01
9	exact_sigma	The <i>exact_sigma</i> denotes the <i>sigma</i> for exact match.	0.001	

## 2.8 ConvKNRM

### 2.8.1 Model Documentation

ConvKNRM Model.

**Examples:**

```
>>> model = ConvKNRM()
>>> model.params['filters'] = 128
>>> model.params['conv_activation_func'] = 'tanh'
>>> model.params['max_ngram'] = 3
>>> model.params['use_crossmatch'] = True
>>> model.params['kernel_num'] = 11
>>> model.params['sigma'] = 0.1
>>> model.params['exact_sigma'] = 0.001
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```



## 2.8.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.conv_knrm.ConvKNRM'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_embedding	Any flag used help <i>auto</i> module. Shouldn't be changed.	True	
3	embedding	FloatTensor containing weights for the Embedding.		
4	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
5	embedding_output_dim	Should be set manually.		
6	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
7	filters	The filter size in the convolution layer.	128	
8	conv_activation	The activation function in the convolution layer.	relu	
9	max_ngram	The maximum length of n-grams for the convolution layer.	3	
10	use_crossmatch	Whether to match left n-grams and right n-grams of different lengths	True	
11	kernel_num	The number of RBF kernels.	11	quantitative uniform distribution in [5, 20), with a step size of 1
12	sigma	The <i>sigma</i> defines the kernel width.	0.1	quantitative uniform distribution in [0.01, 0.2), with a step size of 0.01
13	exact_sigma	The <i>exact_sigma</i> denotes the <i>sigma</i> for exact match.	0.001	

## 2.9 BiMPM

### 2.9.1 Model Documentation

BiMPM Model.

Reference: - <https://github.com/galsang/BIMPM-pytorch/blob/master/model/BIMPM.py>

**Examples:**

```
>>> model = BiMPM()
>>> model.params['num_perspective'] = 4
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.9.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.bimpm.BiMPPM'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_embedding	A flag used help auto module. Shouldn't be changed.	True	
3	embedding	FloatTensor containing weights for the Embedding.		
4	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
5	embedding_output_dim	Should be set manually.		
6	embedding_freeze	True to freeze embedding layer training, False to enable embedding parameters.	False	
7	mask_value	The value to be masked from inputs.	0	
8	dropout	Dropout rate.	0.2	
9	hidden_size	Hidden size.	100	quantitative uniform distribution in [100, 300), with a step size of 100
10	num_perspective	num_perspective	20	quantitative uniform distribution in [20, 100), with a step size of 20

## 2.10 MatchLSTM

### 2.10.1 Model Documentation

MatchLSTM Model.

<https://github.com/shuohangwang/mprc/blob/master/qa/rankerReader.lua>.

**Examples:**

```
>>> model = MatchLSTM()
>>> model.params['dropout'] = 0.2
>>> model.params['hidden_size'] = 200
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.10.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.matchlstm.MatchLSTM'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_embedding	A flag used help <i>auto</i> module. Shouldn't be changed.	True	
3	embedding	FloatTensor containing weights for the Embedding.		
4	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
5	embedding_output_dim	Should be set manually.		
6	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
7	mask_value	The value to be masked from inputs.	0	
8	dropout	Dropout rate.	0.2	
9	hidden_size	Hidden size.	200	
10	lstm_layer	Number of LSTM layers	1	
11	drop_lstm	Whether dropout LSTM.	False	
12	concat_lstm	Whether concat intermediate outputs.	True	
13	rnn_type	Choose rnn type, lstm or gru.	lstm	

## 2.11 ArcII

### 2.11.1 Model Documentation

ArcII Model.

```

Examples:
>>> model = ArcII()
>>> model.params['embedding_output_dim'] = 300
>>> model.params['kernel_1d_count'] = 32
>>> model.params['kernel_1d_size'] = 3
>>> model.params['kernel_2d_count'] = [16, 32]
>>> model.params['kernel_2d_size'] = [[3, 3], [3, 3]]
>>> model.params['pool_2d_size'] = [[2, 2], [2, 2]]
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()

```

## 2.11.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'match-zoo.models.arcii.ArcII'>	
1	task	Decides model output shape, loss, and metrics.		
2	with_embedding	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
3	embedding	FloatTensor containing weights for the Embedding.		
4	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
5	embedding_output_dim	Should be set manually.		
6	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
7	left_length	Length of left input.	8	
8	right_length	Length of right input.	10	
9	kernel_1d_count	Kernel count of 1D convolution layer.	32	
10	kernel_1d_size	Kernel size of 1D convolution layer.	3	
11	kernel_2d_count	Kernel count of 2D convolution layer in each block	[32]	
12	kernel_2d_size	Kernel size of 2D convolution layer in each block.	[(3, 3)]	
13	activation	Activation function.	relu	
14	pool_2d_size	Size of pooling layer in each block.	[(2, 2)]	
15	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

This page contains auto-generated API reference documentation<sup>1</sup>.

## 3.1 matchzoo

### 3.1.1 Subpackages

`matchzoo.auto`

#### Subpackages

`matchzoo.auto.preparer`

#### Submodules

`matchzoo.auto.preparer.prepare`

#### Module Contents

`matchzoo.auto.preparer.prepare.prepare` (*task:BaseTask, model\_class:typing.Type[BaseModel], data\_pack:mz.DataPack, call-back:typing.Optional[BaseCallback]=None, preprocessor:typing.Optional[BasePreprocessor]=None, embedding:typing.Optional['mz.Embedding']=None, config:typing.Optional[dict]=None*)

A simple shorthand for using `matchzoo.Preparer`.

*config* is used to control specific behaviors. The default *config* will be updated accordingly if a *config* dictionary is passed. e.g. to override the default *bin\_size*, pass *config*={ 'bin\_size': 15}.

---

<sup>1</sup> Created with sphinx-autoapi

### Parameters

- **task** – Task.
- **model\_class** – Model class.
- **data\_pack** – DataPack used to fit the preprocessor.
- **callback** – Callback used to padding a batch. (default: the default callback of *model\_class*)
- **preprocessor** – Preprocessor used to fit the *data\_pack*. (default: the default preprocessor of *model\_class*)
- **embedding** – Embedding to build a embedding matrix. If not set, then a correctly shaped randomized matrix will be built.
- **config** – Configuration of specific behaviors. (default: return value of *mz.Preparer.get\_default\_config()*)

**Returns** A tuple of (*model*, *preprocessor*, *data\_generator\_builder*, *embedding\_matrix*).

`matchzoo.auto.preparer.preparer`

### Module Contents

**class** `matchzoo.auto.preparer.preparer.Preparer` (*task:BaseTask*, *config:typing.Optional[dict]=None*)

Bases: `object`

Unified setup processes of all MatchZoo models.

*config* is used to control specific behaviors. The default *config* will be updated accordingly if a *config* dictionary is passed. e.g. to override the default *bin\_size*, pass *config*={*'bin\_size': 15*}.

See *tutorials/automation.ipynb* for a detailed walkthrough on usage.

Default *config*:

```
{ # pair generator builder kwargs 'num_dup': 1,
  # histogram unit of DRMM 'bin_size': 30, 'hist_mode': 'LCH',
  # dynamic Pooling of MatchPyramid 'compress_ratio_left': 1.0, 'compress_ratio_right': 1.0,
  # if no matchzoo.Embedding is passed to tune 'embedding_output_dim': 50
}
```

### Parameters

- **task** – Task.
- **config** – Configuration of specific behaviors.

### Example

```
>>> import matchzoo as mz
>>> task = mz.tasks.Ranking(losses=mz.losses.RankCrossEntropyLoss())
>>> preparer = mz.auto.Preparer(task)
>>> model_class = mz.models.DenseBaseline
```

(continues on next page)

(continued from previous page)

```

>>> train_raw = mz.datasets.toy.load_data('train', 'ranking')
>>> model, prpr, dsb, dlb = preparer.prepare(model_class,
...                                         train_raw)
>>> model.params.completed()
True

```

**prepare** (*self*, *model\_class*:typing.Type[BaseModel], *data\_pack*:mz.DataPack, *callback*:typing.Optional[BaseCallback]=None, *preprocessor*:typing.Optional[BasePreprocessor]=None, *embedding*:typing.Optional['mz.Embedding']=None) Prepare.

#### Parameters

- **model\_class** – Model class.
- **data\_pack** – DataPack used to fit the preprocessor.
- **callback** – Callback used to padding a batch. (default: the default callback of *model\_class*)
- **preprocessor** – Preprocessor used to fit the *data\_pack*. (default: the default preprocessor of *model\_class*)

**Returns** A tuple of (*model*, *preprocessor*, *dataset\_builder*, *dataloader\_builder*).

**\_build\_model** (*self*, *model\_class*, *preprocessor*, *embedding*)

**\_build\_matrix** (*self*, *preprocessor*, *embedding*)

**\_build\_dataset\_builder** (*self*, *model*, *embedding\_matrix*, *preprocessor*)

**\_build\_dataloader\_builder** (*self*, *model*, *callback*)

**\_infer\_num\_neg** (*self*)

**classmethod get\_default\_config** (*cls*)

Default config getter.

## Package Contents

**class** matchzoo.auto.preparer.Preparer (*task*:BaseTask, *config*:typing.Optional[dict]=None)

Bases: object

Unified setup processes of all MatchZoo models.

*config* is used to control specific behaviors. The default *config* will be updated accordingly if a *config* dictionary is passed. e.g. to override the default *bin\_size*, pass *config*={'bin\_size': 15}.

See *tutorials/automation.ipynb* for a detailed walkthrough on usage.

Default *config*:

```

{ # pair generator builder kwargs 'num_dup': 1,
  # histogram unit of DRMM 'bin_size': 30, 'hist_mode': 'LCH',
  # dynamic Pooling of MatchPyramid 'compress_ratio_left': 1.0, 'compress_ratio_right': 1.0,
  # if no matchzoo.Embedding is passed to tune 'embedding_output_dim': 50
}

```

#### Parameters

- **task** – Task.
- **config** – Configuration of specific behaviors.

### Example

```
>>> import matchzoo as mz
>>> task = mz.tasks.Ranking(losses=mz.losses.RankCrossEntropyLoss())
>>> preparer = mz.auto.Preparer(task)
>>> model_class = mz.models.DenseBaseline
>>> train_raw = mz.datasets.toy.load_data('train', 'ranking')
>>> model, prpr, dsb, dlb = preparer.prepare(model_class,
...                                         train_raw)
>>> model.params.completed()
True
```

**prepare** (*self*, *model\_class*:typing.Type[BaseModel], *data\_pack*:mz.DataPack, *callback*:typing.Optional[BaseCallback]=None, *preprocessor*:typing.Optional[BasePreprocessor]=None, *embedding*:typing.Optional['mz.Embedding']=None) Prepare.

#### Parameters

- **model\_class** – Model class.
- **data\_pack** – DataPack used to fit the preprocessor.
- **callback** – Callback used to padding a batch. (default: the default callback of *model\_class*)
- **preprocessor** – Preprocessor used to fit the *data\_pack*. (default: the default preprocessor of *model\_class*)

**Returns** A tuple of (*model*, *preprocessor*, *dataset\_builder*, *dataloader\_builder*).

**\_build\_model** (*self*, *model\_class*, *preprocessor*, *embedding*)

**\_build\_matrix** (*self*, *preprocessor*, *embedding*)

**\_build\_dataset\_builder** (*self*, *model*, *embedding\_matrix*, *preprocessor*)

**\_build\_dataloader\_builder** (*self*, *model*, *callback*)

**\_infer\_num\_neg** (*self*)

**classmethod get\_default\_config** (*cls*)

Default config getter.

```
matchzoo.auto.preparer.prepare (task:BaseTask, model_class:typing.Type[BaseModel],
                                data_pack:mz.DataPack, callback:typing.Optional[BaseCallback]=None,
                                preprocessor:typing.Optional[BasePreprocessor]=None, embedding:typing.Optional['mz.Embedding']=None,
                                config:typing.Optional[dict]=None)
```

A simple shorthand for using `matchzoo.Preparer`.

*config* is used to control specific behaviors. The default *config* will be updated accordingly if a *config* dictionary is passed. e.g. to override the default *bin\_size*, pass *config*={ 'bin\_size': 15}.

#### Parameters

- **task** – Task.



- **model\_class** – Model class.
- **data\_pack** – DataPack used to fit the preprocessor.
- **callback** – Callback used to padding a batch. (default: the default callback of *model\_class*)
- **preprocessor** – Preprocessor used to fit the *data\_pack*. (default: the default preprocessor of *model\_class*)
- **embedding** – Embedding to build a embedding matrix. If not set, then a correctly shaped randomized matrix will be built.
- **config** – Configuration of specific behaviors. (default: return value of *mz.Preparer.get\_default\_config()*)

**Returns** A tuple of (*model*, *preprocessor*, *data\_generator\_builder*, *embedding\_matrix*).

`matchzoo.auto.tuner`

## Submodules

`matchzoo.auto.tuner.tune`

## Module Contents

```
matchzoo.auto.tuner.tune.tune(params: 'mz.ParamTable', optimizer: str = 'adam',
                             trainloader: mz.dataloader.DataLoader = None, validloader: mz.dataloader.DataLoader = None,
                             embedding: np.ndarray = None, fit_kwargs: dict = None, metric: typing.Union[str, BaseMetric] = None,
                             mode: str = 'maximize', num_runs: int = 10, verbose = 1)
```

Tune model hyper-parameters.

A simple shorthand for using `matchzoo.auto.Tuner`.

*model.params.hyper\_space* represents the model's hyper-parameters search space, which is the cross-product of individual hyper parameter's hyper space. When a *Tuner* builds a model, for each hyper parameter in *model.params*, if the hyper-parameter has a hyper-space, then a sample will be taken in the space. However, if the hyper-parameter does not have a hyper-space, then the default value of the hyper-parameter will be used.

See *tutorials/model\_tuning.ipynb* for a detailed walkthrough on usage.

### Parameters

- **params** – A completed parameter table to tune. Usually *model.params* of the desired model to tune. *params.completed()* should be *True*.
- **optimizer** – Str or *Optimizer* class. Optimizer for optimizing model.
- **trainloader** – Training data to use. Should be a *DataLoader*.
- **validloader** – Testing data to use. Should be a *DataLoader*.
- **embedding** – Embedding used by model.
- **fit\_kwargs** – Extra keyword arguments to pass to *fit*. (default: *dict(epochs=10, verbose=0)*)
- **metric** – Metric to tune upon. Must be one of the metrics in *model.params['task'].metrics*. (default: the first metric in *params['task'].metrics*).

- **mode** – Either *maximize* the metric or *minimize* the metric. (default: ‘maximize’)
- **num\_runs** – Number of runs. Each run takes a sample in *params.hyper\_space* and build a model based on the sample. (default: 10)
- **callbacks** – A list of callbacks to handle. Handled sequentially at every callback point.
- **verbose** – Verbosity. (default: 1)

### Example

```
>>> import matchzoo as mz
>>> import numpy as np
>>> train = mz.datasets.toy.load_data('train')
>>> valid = mz.datasets.toy.load_data('dev')
>>> prpr = mz.models.DenseBaseline.get_default_preprocessor()
>>> train = prpr.fit_transform(train, verbose=0)
>>> valid = prpr.transform(valid, verbose=0)
>>> trainset = mz.dataloader.Dataset(train)
>>> validset = mz.dataloader.Dataset(valid)
>>> padding = mz.models.DenseBaseline.get_default_padding_callback()
>>> trainloader = mz.dataloader.DataLoader(trainset, callback=padding)
>>> validloader = mz.dataloader.DataLoader(validset, callback=padding)
>>> model = mz.models.DenseBaseline()
>>> model.params['task'] = mz.tasks.Ranking()
>>> optimizer = 'adam'
>>> embedding = np.random.uniform(-0.2, 0.2,
...                               (prpr.context['vocab_size'], 100))
>>> tuner = mz.auto.Tuner(
...     params=model.params,
...     optimizer=optimizer,
...     trainloader=trainloader,
...     validloader=validloader,
...     embedding=embedding,
...     num_runs=1,
...     verbose=0
... )
>>> results = tuner.tune()
>>> sorted(results['best'].keys())
['#', 'params', 'sample', 'score']
```

`matchzoo.auto.tuner.tuner`

### Module Contents

**class** `matchzoo.auto.tuner.tuner.Tuner` (*params*:‘*mz.ParamTable*’, *optimizer*:str=‘adam’,  
*trainloader*:*mz.dataloader.DataLoader*=None, *valid-*  
*loader*:*mz.dataloader.DataLoader*=None, *embed-*  
*ding*:*np.ndarray*=None, *fit\_kwargs*:dict=None,  
*metric*:*typing.Union[str, BaseMetric]*=None,  
*mode*:str=‘maximize’, *num\_runs*:int=10, *verbose*=1)

Bases: object

Model hyper-parameters tuner.

`model.params.hyper_space` represents the model's hyper-parameters search space, which is the cross-product of individual hyper parameter's hyper space. When a *Tuner* builds a model, for each hyper parameter in `model.params`, if the hyper-parameter has a hyper-space, then a sample will be taken in the space. However, if the hyper-parameter does not have a hyper-space, then the default value of the hyper-parameter will be used.

See `tutorials/model_tuning.ipynb` for a detailed walkthrough on usage.

### Parameters

- **params** – A completed parameter table to tune. Usually `model.params` of the desired model to tune. `params.completed()` should be *True*.
- **optimizer** – Str or *Optimizer* class. Optimizer for optimizing model.
- **trainloader** – Training data to use. Should be a *DataLoader*.
- **validloader** – Testing data to use. Should be a *DataLoader*.
- **embedding** – Embedding used by model.
- **fit\_kwargs** – Extra keyword arguments to pass to *fit*. (default: `dict(epochs=10, verbose=0)`)
- **metric** – Metric to tune upon. Must be one of the metrics in `model.params['task'].metrics`. (default: the first metric in `params['task'].metrics`).
- **mode** – Either *maximize* the metric or *minimize* the metric. (default: 'maximize')
- **num\_runs** – Number of runs. Each run takes a sample in `params.hyper_space` and build a model based on the sample. (default: 10)
- **verbose** – Verbosity. (default: 1)

### params

*params* getter.

### trainloader

*trainloader* getter.

### validloader

*validloader* getter.

### fit\_kwargs

*fit\_kwargs* getter.

### metric

*metric* getter.

### mode

*mode* getter.

### num\_runs

*num\_runs* getter.

### verbose

*verbose* getter.

### tune (self)

Start tuning.

Notice that *tune* does not affect the tuner's inner state, so each new call to *tune* starts fresh. In other words, hyperspaces are suggestive only within the same *tune* call.

### \_fmin (self, trials)

### \_run (self, sample)

```
_create_full_params (self, sample)
_fix_loss_sign (self, loss)
classmethod _log_result (cls, result)
classmethod _validate_params (cls, params)
classmethod _validate_optimizer (cls, optimizer)
classmethod _validate_data_loader (cls, data)
classmethod _validate_kwargs (cls, kwargs)
classmethod _validate_mode (cls, mode)
classmethod _validate_metric (cls, params, metric)
classmethod _validate_num_runs (cls, num_runs)
```

## Package Contents

```
class matchzoo.auto.tuner.Tuner (params:'mz.ParamTable',          optimizer:str='adam',
                                trainloader:mz.dataloader.DataLoader=None, valid-
                                loader:mz.dataloader.DataLoader=None,          embed-
                                ding:np.ndarray=None,          fit_kwargs:dict=None,
                                metric:typing.Union[str,          BaseMetric]=None,
                                mode:str='maximize', num_runs:int=10, verbose=1)
```

Bases: `object`

Model hyper-parameters tuner.

`model.params.hyper_space` represents the model's hyper-parameters search space, which is the cross-product of individual hyper parameter's hyper space. When a *Tuner* builds a model, for each hyper parameter in `model.params`, if the hyper-parameter has a hyper-space, then a sample will be taken in the space. However, if the hyper-parameter does not have a hyper-space, then the default value of the hyper-parameter will be used.

See [tutorials/model\\_tuning.ipynb](#) for a detailed walkthrough on usage.

### Parameters

- **params** – A completed parameter table to tune. Usually `model.params` of the desired model to tune. `params.completed()` should be *True*.
- **optimizer** – Str or *Optimizer* class. Optimizer for optimizing model.
- **trainloader** – Training data to use. Should be a *DataLoader*.
- **validloader** – Testing data to use. Should be a *DataLoader*.
- **embedding** – Embedding used by model.
- **fit\_kwargs** – Extra keyword arguments to pass to *fit*. (default: `dict(epochs=10, verbose=0)`)
- **metric** – Metric to tune upon. Must be one of the metrics in `model.params['task'].metrics`. (default: the first metric in `params['task'].metrics`).
- **mode** – Either *maximize* the metric or *minimize* the metric. (default: 'maximize')
- **num\_runs** – Number of runs. Each run takes a sample in `params.hyper_space` and build a model based on the sample. (default: 10)
- **verbose** – Verbosity. (default: 1)

**params***params* getter.**trainloader***trainloader* getter.**validloader***validloader* getter.**fit\_kwargs***fit\_kwargs* getter.**metric***metric* getter.**mode***mode* getter.**num\_runs***num\_runs* getter.**verbose***verbose* getter.**tune** (*self*)

Start tuning.

Notice that *tune* does not affect the tuner's inner state, so each new call to *tune* starts fresh. In other words, hyperspaces are suggestive only within the same *tune* call.

**\_fmin** (*self*, *trials*)**\_run** (*self*, *sample*)**\_create\_full\_params** (*self*, *sample*)**\_fix\_loss\_sign** (*self*, *loss*)**classmethod \_log\_result** (*cls*, *result*)**classmethod \_validate\_params** (*cls*, *params*)**classmethod \_validate\_optimizer** (*cls*, *optimizer*)**classmethod \_validate\_dataloader** (*cls*, *data*)**classmethod \_validate\_kwargs** (*cls*, *kwargs*)**classmethod \_validate\_mode** (*cls*, *mode*)**classmethod \_validate\_metric** (*cls*, *params*, *metric*)**classmethod \_validate\_num\_runs** (*cls*, *num\_runs*)

```
matchzoo.auto.tuner.tune (params:'mz.ParamTable',          optimizer:str='adam',          train-
                        loader:mz.dataloader.DataLoader=None,          valid-
                        loader:mz.dataloader.DataLoader=None, embedding:np.ndarray=None,
                        fit_kwargs:dict=None, metric:typing.Union[str, BaseMetric]=None,
                        mode:str='maximize', num_runs:int=10, verbose=1)
```

Tune model hyper-parameters.

A simple shorthand for using `matchzoo.auto.Tuner`.

`model.params.hyper_space` represents the model's hyper-parameters search space, which is the cross-product of individual hyper parameter's hyper space. When a *Tuner* builds a model, for each hyper parameter in

*model.params*, if the hyper-parameter has a hyper-space, then a sample will be taken in the space. However, if the hyper-parameter does not have a hyper-space, then the default value of the hyper-parameter will be used.

See *tutorials/model\_tuning.ipynb* for a detailed walkthrough on usage.

### Parameters

- **params** – A completed parameter table to tune. Usually *model.params* of the desired model to tune. *params.completed()* should be *True*.
- **optimizer** – Str or *Optimizer* class. Optimizer for optimizing model.
- **trainloader** – Training data to use. Should be a *DataLoader*.
- **validloader** – Testing data to use. Should be a *DataLoader*.
- **embedding** – Embedding used by model.
- **fit\_kwargs** – Extra keyword arguments to pass to *fit*. (default: *dict(epochs=10, verbose=0)*)
- **metric** – Metric to tune upon. Must be one of the metrics in *model.params['task'].metrics*. (default: the first metric in *params['task'].metrics*).
- **mode** – Either *maximize* the metric or *minimize* the metric. (default: 'maximize')
- **num\_runs** – Number of runs. Each run takes a sample in *params.hyper\_space* and build a model based on the sample. (default: 10)
- **callbacks** – A list of callbacks to handle. Handled sequentially at every callback point.
- **verbose** – Verbosity. (default: 1)

### Example

```
>>> import matchzoo as mz
>>> import numpy as np
>>> train = mz.datasets.toy.load_data('train')
>>> valid = mz.datasets.toy.load_data('dev')
>>> prpr = mz.models.DenseBaseline.get_default_preprocessor()
>>> train = prpr.fit_transform(train, verbose=0)
>>> valid = prpr.transform(valid, verbose=0)
>>> trainset = mz.dataloader.Dataset(train)
>>> validset = mz.dataloader.Dataset(valid)
>>> padding = mz.models.DenseBaseline.get_default_padding_callback()
>>> trainloader = mz.dataloader.DataLoader(trainset, callback=padding)
>>> validloader = mz.dataloader.DataLoader(validset, callback=padding)
>>> model = mz.models.DenseBaseline()
>>> model.params['task'] = mz.tasks.Ranking()
>>> optimizer = 'adam'
>>> embedding = np.random.uniform(-0.2, 0.2,
...                               (prpr.context['vocab_size'], 100))
>>> tuner = mz.auto.Tuner(
...     params=model.params,
...     optimizer=optimizer,
...     trainloader=trainloader,
...     validloader=validloader,
...     embedding=embedding,
...     num_runs=1,
...     verbose=0
... )
```

(continues on next page)

(continued from previous page)

```
>>> results = tuner.tune()
>>> sorted(results['best'].keys())
['#', 'params', 'sample', 'score']
```

## Package Contents

**class** `matchzoo.auto.Preparer` (*task:BaseTask, config:typing.Optional[dict]=None*)

Bases: `object`

Unified setup processes of all MatchZoo models.

*config* is used to control specific behaviors. The default *config* will be updated accordingly if a *config* dictionary is passed. e.g. to override the default *bin\_size*, pass *config*={*'bin\_size': 15*}.

See *tutorials/automation.ipynb* for a detailed walkthrough on usage.

Default *config*:

```
{ # pair generator builder kwargs 'num_dup': 1,
  # histogram unit of DRMM 'bin_size': 30, 'hist_mode': 'LCH',
  # dynamic Pooling of MatchPyramid 'compress_ratio_left': 1.0, 'compress_ratio_right': 1.0,
  # if no matchzoo.Embedding is passed to tune 'embedding_output_dim': 50
}
```

### Parameters

- **task** – Task.
- **config** – Configuration of specific behaviors.

## Example

```
>>> import matchzoo as mz
>>> task = mz.tasks.Ranking(losses=mz.losses.RankCrossEntropyLoss())
>>> preparer = mz.auto.Preparer(task)
>>> model_class = mz.models.DenseBaseline
>>> train_raw = mz.datasets.toy.load_data('train', 'ranking')
>>> model, prpr, dsb, dlb = preparer.prepare(model_class,
...                                         train_raw)
>>> model.params.completed()
True
```

**prepare** (*self*, *model\_class:typing.Type[BaseModel]*, *data\_pack:mz.DataPack*,  
*callback:typing.Optional[BaseCallback]=None*, *preprocessor:typing.Optional[BasePreprocessor]=None*, *embedding:typing.Optional['mz.Embedding']=None*)  
 Prepare.

### Parameters

- **model\_class** – Model class.
- **data\_pack** – DataPack used to fit the preprocessor.
- **callback** – Callback used to padding a batch. (default: the default callback of *model\_class*)

- **preprocessor** – Preprocessor used to fit the *data\_pack*. (default: the default preprocessor of *model\_class*)

**Returns** A tuple of (*model*, *preprocessor*, *dataset\_builder*, *dataloader\_builder*).

**`_build_model`** (*self*, *model\_class*, *preprocessor*, *embedding*)

**`_build_matrix`** (*self*, *preprocessor*, *embedding*)

**`_build_dataset_builder`** (*self*, *model*, *embedding\_matrix*, *preprocessor*)

**`_build_dataloader_builder`** (*self*, *model*, *callback*)

**`_infer_num_neg`** (*self*)

**`classmethod get_default_config`** (*cls*)

Default config getter.

```
class matchzoo.auto.Tuner (params:'mz.ParamTable', optimizer:str='adam',
                           trainloader:mz.dataloader.DataLoader=None, valid-
                           loader:mz.dataloader.DataLoader=None, embed-
                           ding:np.ndarray=None, fit_kwargs:dict=None, metric:typing.Union[str,
                           BaseMetric]=None, mode:str='maximize', num_runs:int=10, ver-
                           bose=1)
```

Bases: object

Model hyper-parameters tuner.

*model.params.hyper\_space* represents the model's hyper-parameters search space, which is the cross-product of individual hyper parameter's hyper space. When a *Tuner* builds a model, for each hyper parameter in *model.params*, if the hyper-parameter has a hyper-space, then a sample will be taken in the space. However, if the hyper-parameter does not have a hyper-space, then the default value of the hyper-parameter will be used.

See *tutorials/model\_tuning.ipynb* for a detailed walkthrough on usage.

### Parameters

- **params** – A completed parameter table to tune. Usually *model.params* of the desired model to tune. *params.completed()* should be *True*.
- **optimizer** – Str or *Optimizer* class. Optimizer for optimizing model.
- **trainloader** – Training data to use. Should be a *DataLoader*.
- **validloader** – Testing data to use. Should be a *DataLoader*.
- **embedding** – Embedding used by model.
- **fit\_kwargs** – Extra keyword arguments to pass to *fit*. (default: *dict(epochs=10, verbose=0)*)
- **metric** – Metric to tune upon. Must be one of the metrics in *model.params['task'].metrics*. (default: the first metric in *params['task'].metrics*).
- **mode** – Either *maximize* the metric or *minimize* the metric. (default: 'maximize')
- **num\_runs** – Number of runs. Each run takes a sample in *params.hyper\_space* and build a model based on the sample. (default: 10)
- **verbose** – Verbosity. (default: 1)

**params**

*params* getter.

**trainloader**

*trainloader* getter.



**validloader**  
*validloader* getter.

**fit\_kwargs**  
*fit\_kwargs* getter.

**metric**  
*metric* getter.

**mode**  
*mode* getter.

**num\_runs**  
*num\_runs* getter.

**verbose**  
*verbose* getter.

**tune** (*self*)  
 Start tuning.

Notice that *tune* does not affect the tuner's inner state, so each new call to *tune* starts fresh. In other words, hyperspaces are suggestive only within the same *tune* call.

**\_fmin** (*self*, *trials*)

**\_run** (*self*, *sample*)

**\_create\_full\_params** (*self*, *sample*)

**\_fix\_loss\_sign** (*self*, *loss*)

**classmethod \_log\_result** (*cls*, *result*)

**classmethod \_validate\_params** (*cls*, *params*)

**classmethod \_validate\_optimizer** (*cls*, *optimizer*)

**classmethod \_validate\_dataloader** (*cls*, *data*)

**classmethod \_validate\_kwargs** (*cls*, *kwargs*)

**classmethod \_validate\_mode** (*cls*, *mode*)

**classmethod \_validate\_metric** (*cls*, *params*, *metric*)

**classmethod \_validate\_num\_runs** (*cls*, *num\_runs*)

**matchzoo.data\_pack**

## Submodules

**matchzoo.data\_pack.data\_pack**

Matchzoo DataPack, pair-wise tuple (feature) and context as input.

## Module Contents

**matchzoo.data\_pack.data\_pack.\_convert\_to\_list\_index** (*index:typing.Union[int, slice, np.array]*, *length:int*)

```
class matchzoo.data_pack.data_pack.DataPack (relation:pd.DataFrame, left:pd.DataFrame,
                                             right:pd.DataFrame)
```

Bases: object

Matchzoo *DataPack* data structure, store dataframe and context.

*DataPack* is a MatchZoo native data structure that most MatchZoo data handling processes build upon. A *DataPack* consists of three parts: *left*, *right* and *relation*, each one of is a *pandas.DataFrame*.

#### Parameters

- **relation** – Store the relation between left document and right document use ids.
- **left** – Store the content or features for id\_left.
- **right** – Store the content or features for id\_right.

#### Example

```
>>> left = [
...     ['qid1', 'query 1'],
...     ['qid2', 'query 2']
... ]
>>> right = [
...     ['did1', 'document 1'],
...     ['did2', 'document 2']
... ]
>>> relation = [['qid1', 'did1', 1], ['qid2', 'did2', 1]]
>>> relation_df = pd.DataFrame(relation)
>>> left = pd.DataFrame(left)
>>> right = pd.DataFrame(right)
>>> dp = DataPack(
...     relation=relation_df,
...     left=left,
...     right=right,
... )
>>> len(dp)
2
```

```
class FrameView (data_pack:'DataPack')
```

Bases: object

FrameView.

```
__getitem__ (self, index:typing.Union[int, slice, np.array])
    Slicer.
```

```
__call__ (self)
    Returns A full copy. Equivalant to frame[:].
```

```
DATA_FILENAME = data.dill
```

```
has_label
```

*True* if *label* column exists, *False* other wise.

**Type** return

```
frame
```

View the data pack as a *pandas.DataFrame*.

Returned data frame is created by merging the left data frame, the right dataframe and the relation data frame. Use *[]* to access an item or a slice of items.

**Returns** A `matchzoo.DataPack.FrameView` instance.

### Example

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> type(data_pack.frame)
<class 'matchzoo.data_pack.data_pack.DataPack.FrameView'>
>>> frame_slice = data_pack.frame[0:5]
>>> type(frame_slice)
<class 'pandas.core.frame.DataFrame'>
>>> list(frame_slice.columns)
['id_left', 'text_left', 'id_right', 'text_right', 'label']
>>> full_frame = data_pack.frame()
>>> len(full_frame) == len(data_pack)
True
```

#### **relation**

*relation* getter.

#### **left**

Get `left()` of `DataPack`.

#### **right**

Get `right()` of `DataPack`.

#### **\_\_len\_\_**(*self*)

Get numer of rows in the class:`DataPack` object.

#### **unpack**(*self*)

Unpack the data for training.

The return value can be directly feed to `model.fit` or `model.fit_generator`.

**Returns** A tuple of (X, y). y is *None* if *self* has no label.

### Example

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> X, y = data_pack.unpack()
>>> type(X)
<class 'dict'>
>>> sorted(X.keys())
['id_left', 'id_right', 'text_left', 'text_right']
>>> type(y)
<class 'numpy.ndarray'>
>>> X, y = data_pack.drop_label().unpack()
>>> type(y)
<class 'NoneType'>
```

#### **\_\_getitem\_\_**(*self*, *index:typing.Union[int, slice, np.array]*)

Get specific item(s) as a new `DataPack`.

The returned `DataPack` will be a copy of the subset of the original `DataPack`.

**Parameters** **index** – Index of the item(s) to get.

**Returns** An instance of *DataPack*.

**copy** (*self*)

**Returns** A deep copy.

**save** (*self*, *dirpath*:*typing.Union[str, Path]*)

Save the *DataPack* object.

A saved *DataPack* is represented as a directory with a *DataPack* object (transformed user input as features and context), it will be saved by *pickle*.

**Parameters** *dirpath* – directory path of the saved *DataPack*.

**\_optional\_inplace** (*func*)

Decorator that adds *inplace* key word argument to a method.

Decorate any method that modifies inplace to make that inplace change optional.

**shuffle** (*self*)

Shuffle the data pack by shuffling the relation column.

**Parameters** *inplace* – *True* to modify inplace, *False* to return a modified copy. (default: *False*)

### Example

```
>>> import matchzoo as mz
>>> import numpy.random
>>> numpy.random.seed(0)
>>> data_pack = mz.datasets.toy.load_data()
>>> orig_ids = data_pack.relation['id_left']
>>> shuffled = data_pack.shuffle()
>>> (shuffled.relation['id_left'] != orig_ids).any()
True
```

**drop\_label** (*self*)

Remove *label* column from the data pack.

**Parameters** *inplace* – *True* to modify inplace, *False* to return a modified copy. (default: *False*)

### Example

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> data_pack.has_label
True
>>> data_pack.drop_label(inplace=True)
>>> data_pack.has_label
False
```

**append\_text\_length** (*self*, *verbose*=1)

Append *length\_left* and *length\_right* columns.

**Parameters**

- **inplace** – *True* to modify inplace, *False* to return a modified copy. (default: *False*)
- **verbose** – Verbosity.

## Example

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> 'length_left' in data_pack.frame[0].columns
False
>>> new_data_pack = data_pack.append_text_length(verbose=0)
>>> 'length_left' in new_data_pack.frame[0].columns
True
>>> 'length_left' in data_pack.frame[0].columns
False
>>> data_pack.append_text_length(inplace=True, verbose=0)
>>> 'length_left' in data_pack.frame[0].columns
True
```

**apply\_on\_text** (*self*, *func*:typing.Callable, *mode*:str='both', *rename*:typing.Optional[str]=None, *verbose*:int=1)

Apply *func* to text columns based on *mode*.

### Parameters

- **func** – The function to apply.
- **mode** – One of “both”, “left” and “right”.
- **rename** – If set, use new names for results instead of replacing the original columns. To set *rename* in “both” mode, use a tuple of *str*, e.g. (“text\_left\_new\_name”, “text\_right\_new\_name”).
- **inplace** – *True* to modify inplace, *False* to return a modified copy. (default: *False*)
- **verbose** – Verbosity.

### Examples::

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> frame = data_pack.frame
```

To apply *len* on the left text and add the result as ‘length\_left’:

```
>>> data_pack.apply_on_text(len, mode='left',
...                         rename='length_left',
...                         inplace=True,
...                         verbose=0)
>>> list(frame[0].columns) # noqa: E501
['id_left', 'text_left', 'length_left', 'id_right', 'text_right', 'label']
```

To do the same to the right text:

```
>>> data_pack.apply_on_text(len, mode='right',
...                         rename='length_right',
...                         inplace=True,
...                         verbose=0)
>>> list(frame[0].columns) # noqa: E501
['id_left', 'text_left', 'length_left', 'id_right', 'text_right', 'length_
↵right', 'label']
```

To do the same to the both texts at the same time:

```
>>> data_pack.apply_on_text(len, mode='both',
...                          rename=('extra_left', 'extra_right'),
...                          inplace=True,
...                          verbose=0)
>>> list(frame[0].columns) # noqa: E501
['id_left', 'text_left', 'length_left', 'extra_left', 'id_right', 'text_
→right', 'length_right', 'extra_right', 'label']
```

To suppress outputs:

```
>>> data_pack.apply_on_text(len, mode='both', verbose=0,
...                          inplace=True)
```

`_apply_on_text_right` (*self*, *func*, *rename*, *verbose=1*)

`_apply_on_text_left` (*self*, *func*, *rename*, *verbose=1*)

`_apply_on_text_both` (*self*, *func*, *rename*, *verbose=1*)

`matchzoo.data_pack.data_pack.load_data_pack` (*dirpath*:*typing.Union[str, Path]*) → *Data-Pack*

Load a *DataPack*. The reverse function of `save()`.

**Parameters** *dirpath* – directory path of the saved model.

**Returns** a *DataPack* instance.

## `matchzoo.data_pack.pack`

Convert list of input into class:*DataPack* expected format.

## Module Contents

`matchzoo.data_pack.pack.pack` (*df*:*pd.DataFrame*) → '*matchzoo.DataPack*'

Pack a *DataPack* using *df*.

The *df* must have *text\_left* and *text\_right* columns. Optionally, the *df* can have *id\_left*, *id\_right* to index *text\_left* and *text\_right* respectively. *id\_left*, *id\_right* will be automatically generated if not specified.

**Parameters** *df* – Input *pandas.DataFrame* to use.

**Examples::**

```
>>> import matchzoo as mz
>>> import pandas as pd
>>> df = pd.DataFrame(data={'text_left': list('AABC'),
...                        'text_right': list('abbc'),
...                        'label': [0, 1, 1, 0]})
>>> mz.pack(df).frame()
   id_left text_left id_right text_right  label
0     L-0         A       R-0         a      0
1     L-0         A       R-1         b      1
2     L-1         B       R-1         b      1
3     L-2         C       R-2         c      0
```

`matchzoo.data_pack.pack._merge` (*data*:*pd.DataFrame*, *ids*:*typing.Union[list, np.array]*, *text\_label*:*str*, *id\_label*:*str*)

```
matchzoo.data_pack.pack._gen_ids (data:pd.DataFrame, col:str, prefix:str)
```

## Package Contents

```
class matchzoo.data_pack.DataPack (relation:pd.DataFrame, left:pd.DataFrame,
                                     right:pd.DataFrame)
```

Bases: object

Matchzoo *DataPack* data structure, store dataframe and context.

*DataPack* is a MatchZoo native data structure that most MatchZoo data handling processes build upon. A *DataPack* consists of three parts: *left*, *right* and *relation*, each one of is a *pandas.DataFrame*.

### Parameters

- **relation** – Store the relation between left document and right document use ids.
- **left** – Store the content or features for id\_left.
- **right** – Store the content or features for id\_right.

## Example

```
>>> left = [
...     ['qid1', 'query 1'],
...     ['qid2', 'query 2']
... ]
>>> right = [
...     ['did1', 'document 1'],
...     ['did2', 'document 2']
... ]
>>> relation = [['qid1', 'did1', 1], ['qid2', 'did2', 1]]
>>> relation_df = pd.DataFrame(relation)
>>> left = pd.DataFrame(left)
>>> right = pd.DataFrame(right)
>>> dp = DataPack(
...     relation=relation_df,
...     left=left,
...     right=right,
... )
>>> len(dp)
2
```

```
class FrameView (data_pack:'DataPack')
```

Bases: object

FrameView.

```
__getitem__ (self, index:typing.Union[int, slice, np.array])
    Slicer.
```

```
__call__ (self)
    Returns A full copy. Equivalant to frame[:].
```

```
DATA_FILENAME = data.dill
```

```
has_label
```

True if *label* column exists, False other wise.

**Type** return

**frame**

View the data pack as a `pandas.DataFrame`.

Returned data frame is created by merging the left data frame, the right dataframe and the relation data frame. Use `[]` to access an item or a slice of items.

**Returns** A `matchzoo.DataPack.FrameView` instance.

**Example**

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> type(data_pack.frame)
<class 'matchzoo.data_pack.data_pack.DataPack.FrameView'>
>>> frame_slice = data_pack.frame[0:5]
>>> type(frame_slice)
<class 'pandas.core.frame.DataFrame'>
>>> list(frame_slice.columns)
['id_left', 'text_left', 'id_right', 'text_right', 'label']
>>> full_frame = data_pack.frame()
>>> len(full_frame) == len(data_pack)
True
```

**relation**

*relation* getter.

**left**

Get `left()` of `DataPack`.

**right**

Get `right()` of `DataPack`.

**\_\_len\_\_ (self)**

Get numer of rows in the class:`DataPack` object.

**unpack (self)**

Unpack the data for training.

The return value can be directly feed to `model.fit` or `model.fit_generator`.

**Returns** A tuple of (X, y). y is `None` if `self` has no label.

**Example**

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> X, y = data_pack.unpack()
>>> type(X)
<class 'dict'>
>>> sorted(X.keys())
['id_left', 'id_right', 'text_left', 'text_right']
>>> type(y)
<class 'numpy.ndarray'>
>>> X, y = data_pack.drop_label().unpack()
>>> type(y)
<class 'NoneType'>
```



`__getitem__` (*self*, *index*:`typing.Union[int, slice, np.array]`)

Get specific item(s) as a new *DataPack*.

The returned *DataPack* will be a copy of the subset of the original *DataPack*.

**Parameters** *index* – Index of the item(s) to get.

**Returns** An instance of *DataPack*.

`copy` (*self*)

**Returns** A deep copy.

`save` (*self*, *dirpath*:`typing.Union[str, Path]`)

Save the *DataPack* object.

A saved *DataPack* is represented as a directory with a *DataPack* object (transformed user input as features and context), it will be saved by *pickle*.

**Parameters** *dirpath* – directory path of the saved *DataPack*.

`_optional_inplace` (*func*)

Decorator that adds *inplace* key word argument to a method.

Decorate any method that modifies inplace to make that inplace change optional.

`shuffle` (*self*)

Shuffle the data pack by shuffling the relation column.

**Parameters** *inplace* – *True* to modify inplace, *False* to return a modified copy. (default: *False*)

### Example

```
>>> import matchzoo as mz
>>> import numpy.random
>>> numpy.random.seed(0)
>>> data_pack = mz.datasets.toy.load_data()
>>> orig_ids = data_pack.relation['id_left']
>>> shuffled = data_pack.shuffle()
>>> (shuffled.relation['id_left'] != orig_ids).any()
True
```

`drop_label` (*self*)

Remove *label* column from the data pack.

**Parameters** *inplace* – *True* to modify inplace, *False* to return a modified copy. (default: *False*)

### Example

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> data_pack.has_label
True
>>> data_pack.drop_label(inplace=True)
>>> data_pack.has_label
False
```

**append\_text\_length** (*self*, *verbose=1*)  
Append *length\_left* and *length\_right* columns.

#### Parameters

- **inplace** – *True* to modify inplace, *False* to return a modified copy. (default: *False*)
- **verbose** – Verbosity.

#### Example

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> 'length_left' in data_pack.frame[0].columns
False
>>> new_data_pack = data_pack.append_text_length(verbose=0)
>>> 'length_left' in new_data_pack.frame[0].columns
True
>>> 'length_left' in data_pack.frame[0].columns
False
>>> data_pack.append_text_length(inplace=True, verbose=0)
>>> 'length_left' in data_pack.frame[0].columns
True
```

**apply\_on\_text** (*self*, *func:typing.Callable*, *mode:str='both'*, *rename:typing.Optional[str]=None*,  
*verbose:int=1*)  
Apply *func* to text columns based on *mode*.

#### Parameters

- **func** – The function to apply.
- **mode** – One of “both”, “left” and “right”.
- **rename** – If set, use new names for results instead of replacing the original columns. To set *rename* in “both” mode, use a tuple of *str*, e.g. (“text\_left\_new\_name”, “text\_right\_new\_name”).
- **inplace** – *True* to modify inplace, *False* to return a modified copy. (default: *False*)
- **verbose** – Verbosity.

#### Examples::

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> frame = data_pack.frame
```

To apply *len* on the left text and add the result as ‘length\_left’:

```
>>> data_pack.apply_on_text(len, mode='left',
...                          rename='length_left',
...                          inplace=True,
...                          verbose=0)
>>> list(frame[0].columns) # noqa: E501
['id_left', 'text_left', 'length_left', 'id_right', 'text_right', 'label']
```

To do the same to the right text:

```
>>> data_pack.apply_on_text(len, mode='right',
...                          rename='length_right',
...                          inplace=True,
...                          verbose=0)
>>> list(frame[0].columns) # noqa: E501
['id_left', 'text_left', 'length_left', 'id_right', 'text_right', 'length_
↪right', 'label']
```

To do the same to the both texts at the same time:

```
>>> data_pack.apply_on_text(len, mode='both',
...                          rename=('extra_left', 'extra_right'),
...                          inplace=True,
...                          verbose=0)
>>> list(frame[0].columns) # noqa: E501
['id_left', 'text_left', 'length_left', 'extra_left', 'id_right', 'text_
↪right', 'length_right', 'extra_right', 'label']
```

To suppress outputs:

```
>>> data_pack.apply_on_text(len, mode='both', verbose=0,
...                          inplace=True)
```

`_apply_on_text_right` (*self*, *func*, *rename*, *verbose=1*)

`_apply_on_text_left` (*self*, *func*, *rename*, *verbose=1*)

`_apply_on_text_both` (*self*, *func*, *rename*, *verbose=1*)

`matchzoo.data_pack.load_data_pack` (*dirpath:typing.Union[str, Path]*) → `DataPack`

Load a `DataPack`. The reverse function of `save()`.

**Parameters** `dirpath` – directory path of the saved model.

**Returns** a `DataPack` instance.

`matchzoo.data_pack.pack` (*df:pd.DataFrame*) → `'matchzoo.DataPack'`

Pack a `DataPack` using *df*.

The *df* must have `text_left` and `text_right` columns. Optionally, the *df* can have `id_left`, `id_right` to index `text_left` and `text_right` respectively. `id_left`, `id_right` will be automatically generated if not specified.

**Parameters** `df` – Input `pandas.DataFrame` to use.

**Examples::**

```
>>> import matchzoo as mz
>>> import pandas as pd
>>> df = pd.DataFrame(data={'text_left': list('AABC'),
...                          'text_right': list('abbc'),
...                          'label': [0, 1, 1, 0]})
>>> mz.pack(df).frame()
  id_left text_left id_right text_right  label
0     L-0         A      R-0         a      0
1     L-0         A      R-1         b      1
2     L-1         B      R-1         b      1
3     L-2         C      R-2         c      0
```

`matchzoo.dataloader`

## Subpackages

`matchzoo.dataloader.callbacks`

## Submodules

`matchzoo.dataloader.callbacks.dynamic_pooling`

## Module Contents

**class** `matchzoo.dataloader.callbacks.dynamic_pooling.DynamicPooling` (*fixed\_length\_left:int, fixed\_length\_right:int, compress\_ratio\_left:float=1, compress\_ratio\_right:float=1*)

Bases: `matchzoo.engine.base_callback.BaseCallback`

DPoolPairDataGenerator constructor.

### Parameters

- **fixed\_length\_left** – max length of left text.
- **fixed\_length\_right** – max length of right text.
- **compress\_ratio\_left** – the length change ratio, especially after normal pooling layers.
- **compress\_ratio\_right** – the length change ratio, especially after normal pooling layers.

**on\_batch\_unpacked** (*self, x, y*)  
Insert *dpool\_index* into *x*.

### Parameters

- **x** – unpacked x.
- **y** – unpacked y.

`matchzoo.dataloader.callbacks.dynamic_pooling._dynamic_pooling_index` (*length\_left:np.array, length\_right:np.array, fixed\_length\_left:int, fixed\_length\_right:int, compress\_ratio\_left:float, compress\_ratio\_right:float*)  
→  
`np.array`

`matchzoo.dataloader.callbacks.histogram`

## Module Contents

**class** `matchzoo.dataloader.callbacks.histogram.Histogram` (*embedding\_matrix:np.ndarray,*  
*bin\_size:int=30,*  
*hist\_mode:str='CH'*)

Bases: `matchzoo.engine.base_callback.BaseCallback`

Generate data with matching histogram.

### Parameters

- **embedding\_matrix** – The embedding matrix used to generator match histogram.
- **bin\_size** – The number of bin size of the histogram.
- **hist\_mode** – The mode of the MatchingHistogramUnit, one of *CH*, *NH*, and *LCH*.

**on\_batch\_unpacked** (*self, x, y*)  
 Insert *match\_histogram* to *x*.

`matchzoo.dataloader.callbacks.histogram._trunc_text` (*input\_text:list, length:list*) → list  
 Truncating the input text according to the input length.

### Parameters

- **input\_text** – The input text need to be truncated.
- **length** – The length used to truncated the text.

**Returns** The truncated text.

`matchzoo.dataloader.callbacks.histogram._build_match_histogram` (*x:dict,*  
*match\_hist\_unit:mz.preprocessors.units.l*  
 → np.ndarray)

Generate the matching hisogram for input.

### Parameters

- **x** – The input *dict*.
- **match\_hist\_unit** – The histogram unit MatchingHistogramUnit.

**Returns** The matching histogram.

`matchzoo.dataloader.callbacks.lambda_callback`

## Module Contents

**class** `matchzoo.dataloader.callbacks.lambda_callback.LambdaCallback` (*on\_batch\_data\_pack=None,*  
*on\_batch\_unpacked=None*)

Bases: `matchzoo.engine.base_callback.BaseCallback`

LambdaCallback. Just a shorthand for creating a callback class.

See `matchzoo.engine.base_callback.BaseCallback` for more details.

## Example

```
>>> import matchzoo as mz
>>> from matchzoo.dataloader.callbacks import LambdaCallback
>>> data = mz.datasets.toy.load_data()
>>> batch_func = lambda x: print(type(x))
>>> unpack_func = lambda x, y: print(type(x), type(y))
>>> callback = LambdaCallback(on_batch_data_pack=batch_func,
...                           on_batch_unpacked=unpack_func)
>>> dataset = mz.dataloader.Dataset(
...     data, callbacks=[callback])
>>> _ = dataset[0]
<class 'matchzoo.data_pack.data_pack.DataPack'>
<class 'dict'> <class 'numpy.ndarray'>
```

```
on_batch_data_pack(self, data_pack)
    on_batch_data_pack.
```

```
on_batch_unpacked(self, x, y)
    on_batch_unpacked.
```

## matchzoo.dataloader.callbacks.padding

### Module Contents

```
class matchzoo.dataloader.callbacks.padding.BasicPadding (fixed_length_left:int=None,
                                                         fixed_length_right:int=None,
                                                         pad_value:typing.Union[int,
                                                         str]=0,
                                                         pad_mode:str='pre')
```

Bases: *matchzoo.engine.base\_callback.BaseCallback*

Pad data for basic preprocessor.

#### Parameters

- **fixed\_length\_left** – Integer. If set, *text\_left* will be padded to this length.
- **fixed\_length\_right** – Integer. If set, *text\_right* will be padded to this length.
- **pad\_value** – the value to fill text.
- **pad\_mode** – String, *pre* or *post*: pad either before or after each sequence.

```
on_batch_unpacked(self, x:dict, y:np.ndarray)
    Pad x['text_left'] and x['text_right'].
```

```
class matchzoo.dataloader.callbacks.padding.DRMMPadding (fixed_length_left:int=None,
                                                         fixed_length_right:int=None,
                                                         pad_value:typing.Union[int,
                                                         str]=0,
                                                         pad_mode:str='pre')
```

Bases: *matchzoo.engine.base\_callback.BaseCallback*

Pad data for DRMM Model.

#### Parameters

- **fixed\_length\_left** – Integer. If set, *text\_left* and *match\_histogram* will be padded to this length.
- **fixed\_length\_right** – Integer. If set, *text\_right* will be padded to this length.

- **pad\_value** – the value to fill text.
- **pad\_mode** – String, *pre* or *post*: pad either before or after each sequence.

**on\_batch\_unpacked** (*self*, *x*:dict, *y*:np.ndarray)  
Padding.

Pad *x*['text\_left'], *x*['text\_right'] and *x*['match\_histogram'].

```
class matchzoo.dataloader.callbacks.padding.CDSSMPadding (fixed_length_left:int=None,
                                                         fixed_length_right:int=None,
                                                         pad_value:typing.Union[int,
                                                         str]=0,
                                                         pad_mode:str='pre')
```

Bases: *matchzoo.engine.base\_callback.BaseCallback*

Pad data for cdssm preprocessor.

#### Parameters

- **fixed\_length\_left** – Integer. If set, *text\_left* will be padded to this length.
- **fixed\_length\_right** – Integer. If set, *text\_right* will be padded to this length.
- **pad\_value** – the value to fill text.
- **pad\_mode** – String, *pre* or *post*: pad either before or after each sequence.

**on\_batch\_unpacked** (*self*, *x*:dict, *y*:np.ndarray)  
Pad *x*['text\_left'] and *x*['text\_right'].

```
class matchzoo.dataloader.callbacks.padding.DIINPadding (fixed_length_left:int=None,
                                                         fixed_length_right:int=None,
                                                         fixed_length_word:int=None,
                                                         pad_value:typing.Union[int,
                                                         str]=0,
                                                         pad_mode:str='pre')
```

Bases: *matchzoo.engine.base\_callback.BaseCallback*

Pad data for diin preprocessor.

#### Parameters

- **fixed\_length\_left** – Integer. If set, *text\_left*, *char\_left* and *match\_left* will be padded to this length.
- **fixed\_length\_right** – Integer. If set, *text\_right*, *char\_right* and *match\_right* will be padded to this length.
- **fixed\_length\_word** – Integer. If set, words in *char\_left* and *char\_right* will be padded to this length.
- **pad\_value** – the value to fill text.
- **pad\_mode** – String, *pre* or *post*: pad either before or after each sequence.

**on\_batch\_unpacked** (*self*, *x*:dict, *y*:np.ndarray)  
Padding.

Pad *x*['text\_left'], *x*['text\_right'], *x*['char\_left'], *x*['char\_right'], *x*['match\_left'], *x*['match\_right'].

```
class matchzoo.dataloader.callbacks.padding.BertPadding (fixed_length_left:int=None,  
fixed_length_right:int=None,  
pad_value:typing.Union[int,  
str]=0,  
pad_mode:str='pre')
```

Bases: `matchzoo.engine.base_callback.BaseCallback`

Pad data for bert preprocessor.

#### Parameters

- **fixed\_length\_left** – Integer. If set, `text_left` will be padded to this length.
- **fixed\_length\_right** – Integer. If set, `text_right` will be padded to this length.
- **pad\_value** – the value to fill text.
- **pad\_mode** – String, `pre` or `post`: pad either before or after each sequence.

**on\_batch\_unpacked** (*self, x:dict, y:np.ndarray*)  
Pad `x['text_left']` and `x['text_right']`.

## Package Contents

```
class matchzoo.dataloader.callbacks.LambdaCallback (on_batch_data_pack=None,  
on_batch_unpacked=None)
```

Bases: `matchzoo.engine.base_callback.BaseCallback`

LambdaCallback. Just a shorthand for creating a callback class.

See `matchzoo.engine.base_callback.BaseCallback` for more details.

## Example

```
>>> import matchzoo as mz
>>> from matchzoo.dataloader.callbacks import LambdaCallback
>>> data = mz.datasets.toy.load_data()
>>> batch_func = lambda x: print(type(x))
>>> unpack_func = lambda x, y: print(type(x), type(y))
>>> callback = LambdaCallback(on_batch_data_pack=batch_func,
...                           on_batch_unpacked=unpack_func)
>>> dataset = mz.dataloader.Dataset(
...     data, callbacks=[callback])
>>> _ = dataset[0]
<class 'matchzoo.data_pack.data_pack.DataPack'>
<class 'dict'> <class 'numpy.ndarray'>
```

**on\_batch\_data\_pack** (*self, data\_pack*)  
`on_batch_data_pack`.

**on\_batch\_unpacked** (*self, x, y*)  
`on_batch_unpacked`.

```
class matchzoo.dataloader.callbacks.DynamicPooling (fixed_length_left:int,  
fixed_length_right:int, com-  
press_ratio_left:float=1, com-  
press_ratio_right:float=1)
```

Bases: `matchzoo.engine.base_callback.BaseCallback`

DPoolPairDataGenerator constructor.



**Parameters**

- **fixed\_length\_left** – max length of left text.
- **fixed\_length\_right** – max length of right text.
- **compress\_ratio\_left** – the length change ratio, especially after normal pooling layers.
- **compress\_ratio\_right** – the length change ratio, especially after normal pooling layers.

**on\_batch\_unpacked** (*self*, *x*, *y*)

Insert *dpool\_index* into *x*.

**Parameters**

- **x** – unpacked *x*.
- **y** – unpacked *y*.

```
class matchzoo.dataloader.callbacks.Histogram (embedding_matrix:np.ndarray,
                                             bin_size:int=30, hist_mode:str='CH')
```

Bases: *matchzoo.engine.base\_callback.BaseCallback*

Generate data with matching histogram.

**Parameters**

- **embedding\_matrix** – The embedding matrix used to generator match histogram.
- **bin\_size** – The number of bin size of the histogram.
- **hist\_mode** – The mode of the MatchingHistogramUnit, one of *CH*, *NH*, and *LCH*.

**on\_batch\_unpacked** (*self*, *x*, *y*)

Insert *match\_histogram* to *x*.

```
class matchzoo.dataloader.callbacks.BasicPadding (fixed_length_left:int=None,
                                                  fixed_length_right:int=None,
                                                  pad_value:typing.Union[int, str]=0,
                                                  pad_mode:str='pre')
```

Bases: *matchzoo.engine.base\_callback.BaseCallback*

Pad data for basic preprocessor.

**Parameters**

- **fixed\_length\_left** – Integer. If set, *text\_left* will be padded to this length.
- **fixed\_length\_right** – Integer. If set, *text\_right* will be padded to this length.
- **pad\_value** – the value to fill text.
- **pad\_mode** – String, *pre* or *post*: pad either before or after each sequence.

**on\_batch\_unpacked** (*self*, *x:dict*, *y:np.ndarray*)

Pad *x*['*text\_left*'] and *x*['*text\_right*'].

```
class matchzoo.dataloader.callbacks.DRMMPadding (fixed_length_left:int=None,
                                                  fixed_length_right:int=None,
                                                  pad_value:typing.Union[int, str]=0,
                                                  pad_mode:str='pre')
```

Bases: *matchzoo.engine.base\_callback.BaseCallback*

Pad data for DRMM Model.

**Parameters**

- **fixed\_length\_left** – Integer. If set, *text\_left* and *match\_histogram* will be padded to this length.
- **fixed\_length\_right** – Integer. If set, *text\_right* will be padded to this length.
- **pad\_value** – the value to fill text.
- **pad\_mode** – String, *pre* or *post*: pad either before or after each sequence.

**on\_batch\_unpacked** (*self*, *x*:dict, *y*:np.ndarray)

Padding.

Pad *x*['*text\_left*'], *x*['*text\_right*'] and *x*['*match\_histogram*'].

```
class matchzoo.dataloader.callbacks.CDSSMPadding (fixed_length_left:int=None,
                                                  fixed_length_right:int=None,
                                                  pad_value:typing.Union[int, str]=0,
                                                  pad_mode:str='pre')
```

Bases: *matchzoo.engine.base\_callback.BaseCallback*

Pad data for cdssm preprocessor.

#### Parameters

- **fixed\_length\_left** – Integer. If set, *text\_left* will be padded to this length.
- **fixed\_length\_right** – Integer. If set, *text\_right* will be padded to this length.
- **pad\_value** – the value to fill text.
- **pad\_mode** – String, *pre* or *post*: pad either before or after each sequence.

**on\_batch\_unpacked** (*self*, *x*:dict, *y*:np.ndarray)

Pad *x*['*text\_left*'] and *x*['*text\_right*'].

```
class matchzoo.dataloader.callbacks.DIINPadding (fixed_length_left:int=None,
                                                  fixed_length_right:int=None,
                                                  fixed_length_word:int=None,
                                                  pad_value:typing.Union[int, str]=0,
                                                  pad_mode:str='pre')
```

Bases: *matchzoo.engine.base\_callback.BaseCallback*

Pad data for diin preprocessor.

#### Parameters

- **fixed\_length\_left** – Integer. If set, *text\_left*, *char\_left* and *match\_left* will be padded to this length.
- **fixed\_length\_right** – Integer. If set, *text\_right*, *char\_right* and *match\_right* will be padded to this length.
- **fixed\_length\_word** – Integer. If set, words in *char\_left* and *char\_right* will be padded to this length.
- **pad\_value** – the value to fill text.
- **pad\_mode** – String, *pre* or *post*: pad either before or after each sequence.

**on\_batch\_unpacked** (*self*, *x*:dict, *y*:np.ndarray)

Padding.

Pad *x*['*text\_left*'], *x*['*text\_right*'], *x*['*char\_left*'], *x*['*char\_right*'], *x*['*match\_left*'], *x*['*match\_right*'].

```
class matchzoo.dataloader.callbacks.BertPadding (fixed_length_left:int=None,  
                                              fixed_length_right:int=None,  
                                              pad_value:typing.Union[int, str]=0,  
                                              pad_mode:str='pre')
```

Bases: `matchzoo.engine.base_callback.BaseCallback`

Pad data for bert preprocessor.

#### Parameters

- **fixed\_length\_left** – Integer. If set, *text\_left* will be padded to this length.
- **fixed\_length\_right** – Integer. If set, *text\_right* will be padded to this length.
- **pad\_value** – the value to fill text.
- **pad\_mode** – String, *pre* or *post*: pad either before or after each sequence.

```
on_batch_unpacked (self, x:dict, y:np.ndarray)  
    Pad x['text_left'] and x['text_right'].
```

## Submodules

`matchzoo.dataloader.dataloader`

Basic data loader.

## Module Contents

```
class matchzoo.dataloader.dataloader.DataLoader (dataset:data.Dataset,  
                                              batch_size:int=32, device:  
                                              typing.Optional[torch.device]=None,  
                                              stage='train', resample:bool=True,  
                                              shuffle:bool=False, sort:bool=True,  
                                              callback:BaseCallback=None,  
                                              pin_memory:bool=False, time-  
                                              out:int=0, num_workers:int=0,  
                                              worker_init_fn=None)
```

Bases: `object`

`DataLoader` that loads batches of data from a `Dataset`.

#### Parameters

- **dataset** – The `Dataset` object to load data from.
- **batch\_size** – Batch\_size. (default: 32)
- **device** – An instance of `torch.device` specifying which device the Variables are going to be created on.
- **stage** – One of “train”, “dev”, and “test”. (default: “train”)
- **resample** – Whether to resample data between epochs. only effective when *mode* of dataset is “pair”. (default: *True*)
- **shuffle** – Whether to shuffle data between epochs. (default: *False*)
- **sort** – Whether to sort data according to *length\_right*. (default: *True*)

- **callback** – BaseCallback. See *matchzoo.engine.base\_callback.BaseCallback* for more details.
- **pin\_memory** – If set to *True*, tensors will be copied into pinned memory. (default: *False*)
- **timeout** – The timeout value for collecting a batch from workers. ( default: 0)
- **num\_workers** – The number of subprocesses to use for data loading. 0 means that the data will be loaded in the main process. (default: 0)
- **worker\_init\_fn** – If not *None*, this will be called on each worker subprocess with the worker id (an int in [0, num\_workers - 1]) as input, after seeding and before data loading. (default: *None*)

## Examples

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data(stage='train')
>>> preprocessor = mz.preprocessors.CDSSMPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset = mz.dataloader.Dataset(data_processed, mode='point')
>>> padding_callback = mz.dataloader.callbacks.CDSSMPadding()
>>> dataloader = mz.dataloader.DataLoader(
...     dataset, stage='train', callback=padding_callback)
>>> len(dataloader)
4
```

**id\_left**  
*id\_left* getter.

**label**  
*label* getter.

**\_\_len\_\_**(*self*)  
Get the total number of batches.

**init\_epoch**(*self*)  
Resample, shuffle or sort the dataset for a new epoch.

**\_\_iter\_\_**(*self*)  
Iteration.

**\_handle\_callbacks\_on\_batch\_unpacked**(*self*, *x*, *y*)

**matchzoo.dataloader.dataloader.mz\_collate**(*batch*)  
Put each data field into an array with outer dimension batch size.

**matchzoo.dataloader.dataloader\_builder**

## Module Contents

**class** matchzoo.dataloader.dataloader\_builder.**DataLoaderBuilder**(*\*\*kwargs*)  
Bases: object  
DataLoader Bulider. In essense a wrapped partial function.

## Example

```
>>> import matchzoo as mz
>>> padding_callback = mz.dataloader.callbacks.CDSSMPadding()
>>> builder = mz.dataloader.DataLoaderBuilder(
...     stage='train', callback=padding_callback
... )
>>> data_pack = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.CDSSMPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset = mz.dataloader.Dataset(data_processed, mode='point')
>>> dataloader = builder.build(dataset)
>>> type(dataloader)
<class 'matchzoo.dataloader.dataloader.DataLoader'>
```

**build** (*self*, *dataset*, *\*\*kwargs*)  
Build a DataLoader.

### Parameters

- **dataset** – Dataset to build upon.
- **kwargs** – Additional keyword arguments to override the keyword arguments passed in `__init__`.

## matchzoo.dataloader.dataset

A basic class representing a Dataset.

## Module Contents

```
class matchzoo.dataloader.dataset.Dataset (data_pack:mz.DataPack, mode='point',  
                                           num_dup:int=1, num_neg:int=1, call-  
                                           backs:typing.List[BaseCallback]=None)
```

Bases: `torch.utils.data.Dataset`

Dataset that is built from a data pack.

### Parameters

- **data\_pack** – DataPack to build the dataset.
- **mode** – One of “point”, “pair”, and “list”. (default: “point”)
- **num\_dup** – Number of duplications per instance, only effective when *mode* is “pair”. (default: 1)
- **num\_neg** – Number of negative samples per instance, only effective when *mode* is “pair”. (default: 1)
- **callbacks** – Callbacks. See *matchzoo.data\_generator.callbacks* for more details.

## Examples

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data(stage='train')
>>> preprocessor = mz.preprocessors.CDSSMPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset_point = mz.dataloader.Dataset(data_processed, mode='point')
>>> len(dataset_point)
100
>>> dataset_pair = mz.dataloader.Dataset(
...     data_processed, mode='pair', num_neg=2)
>>> len(dataset_pair)
5
```

**data\_pack**

*data\_pack* getter.

**callbacks**

*callbacks* getter.

**num\_neg**

*num\_neg* getter.

**num\_dup**

*num\_dup* getter.

**mode**

*mode* getter.

**index\_pool**

*index\_pool* getter.

**\_\_len\_\_**(*self*)

Get the total number of instances.

**\_\_getitem\_\_**(*self*, *item:int*)

Get a set of instances from index idx.

Parameters **item** – the index of the instance.

**\_handle\_callbacks\_on\_batch\_data\_pack**(*self*, *batch\_data\_pack*)**\_handle\_callbacks\_on\_batch\_unpacked**(*self*, *x*, *y*)**get\_index\_pool**(*self*)

Set the:attr: *index\_pool*.

Here the *\_index\_pool* records the index of all the instances.

**sample**(*self*)

Resample the instances from data pack.

**shuffle**(*self*)

Shuffle the instances.

**sort**(*self*)

Sort the instances by *length\_right*.

**classmethod \_reorganize\_pair\_wise**(*cls*, *relation:pd.DataFrame*, *num\_dup:int=1*,  
*num\_neg:int=1*)

Re-organize the data pack as pair-wise format.

## matchzoo.dataloader.dataset\_builder

### Module Contents

**class** matchzoo.dataloader.dataset\_builder.**DatasetBuilder** (\*\*kwargs)

Bases: object

Dataset Bulider. In essense a wrapped partial function.

### Example

```

>>> import matchzoo as mz
>>> builder = mz.dataloader.DatasetBuilder(
...     mode='point'
... )
>>> data = mz.datasets.toy.load_data()
>>> gen = builder.build(data)
>>> type(gen)
<class 'matchzoo.dataloader.dataset.Dataset'>

```

**build** (self, data\_pack, \*\*kwargs)

Build a Dataset.

#### Parameters

- **data\_pack** – DataPack to build upon.
- **kwargs** – Additional keyword arguments to override the keyword arguments passed in `__init__`.

## matchzoo.dataloader.sampler

Sampler class for dataloader.

### Module Contents

**class** matchzoo.dataloader.sampler.**SequentialSampler** (dataset:Dataset)

Bases: torch.utils.data.Sampler

Samples elements sequentially, always in the same order.

**Parameters** **dataset** – The dataset to sample from.

**\_\_iter\_\_** (self)

Get the indices of a batch.

**\_\_len\_\_** (self)

Get the total number of instances.

**class** matchzoo.dataloader.sampler.**SortedSampler** (dataset:Dataset)

Bases: torch.utils.data.Sampler

Samples elements according to *length\_right*.

**Parameters** **dataset** – The dataset to sample from.

`__iter__(self)`

Get the indices of a batch.

`__len__(self)`

Get the total number of instances.

**class** `matchzoo.dataloader.sampler.RandomSampler` (*dataset:Dataset*)

Bases: `torch.utils.data.Sampler`

Samples elements randomly.

**Parameters** **dataset** – The dataset to sample from.

`__iter__(self)`

Get the indices of a batch.

`__len__(self)`

Get the total number of instances.

**class** `matchzoo.dataloader.sampler.BatchSampler` (*sampler:Sampler, batch\_size:int=32*)

Bases: `torch.utils.data.Sampler`

Wraps another sampler to yield the indices of a batch.

**Parameters**

- **sampler** – Base sampler.
- **batch\_size** – Size of a batch.

`__iter__(self)`

Get the indices of a batch.

`__len__(self)`

Get the total number of batch.

## Package Contents

**class** `matchzoo.dataloader.Dataset` (*data\_pack:mz.DataPack, mode='point', num\_dup:int=1, num\_neg:int=1, callbacks:typing.List[BaseCallback]=None*)

Bases: `torch.utils.data.Dataset`

Dataset that is built from a data pack.

**Parameters**

- **data\_pack** – DataPack to build the dataset.
- **mode** – One of “point”, “pair”, and “list”. (default: “point”)
- **num\_dup** – Number of duplications per instance, only effective when *mode* is “pair”. (default: 1)
- **num\_neg** – Number of negative samples per instance, only effective when *mode* is “pair”. (default: 1)
- **callbacks** – Callbacks. See `matchzoo.data_generator.callbacks` for more details.



## Examples

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data(stage='train')
>>> preprocessor = mz.preprocessors.CDSSMPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset_point = mz.dataloader.Dataset(data_processed, mode='point')
>>> len(dataset_point)
100
>>> dataset_pair = mz.dataloader.Dataset(
...     data_processed, mode='pair', num_neg=2)
>>> len(dataset_pair)
5
```

**data\_pack**

*data\_pack* getter.

**callbacks**

*callbacks* getter.

**num\_neg**

*num\_neg* getter.

**num\_dup**

*num\_dup* getter.

**mode**

*mode* getter.

**index\_pool**

*index\_pool* getter.

**\_\_len\_\_** (*self*)

Get the total number of instances.

**\_\_getitem\_\_** (*self*, *item:int*)

Get a set of instances from index idx.

**Parameters** *item* – the index of the instance.

**\_handle\_callbacks\_on\_batch\_data\_pack** (*self*, *batch\_data\_pack*)

**\_handle\_callbacks\_on\_batch\_unpacked** (*self*, *x*, *y*)

**get\_index\_pool** (*self*)

Set the:attr: *\_index\_pool*.

Here the *\_index\_pool* records the index of all the instances.

**sample** (*self*)

Resample the instances from data pack.

**shuffle** (*self*)

Shuffle the instances.

**sort** (*self*)

Sort the instances by *length\_right*.

**classmethod \_reorganize\_pair\_wise** (*cls*, *relation:pd.DataFrame*, *num\_dup:int=1*,  
*num\_neg:int=1*)

Re-organize the data pack as pair-wise format.

```
class matchzoo.dataloader.DataLoader (dataset:data.Dataset,      batch_size:int=32,      de-
                                     vice:typing.Optional[torch.device]=None,
                                     stage='train', resample:bool=True, shuffle:bool=False,
                                     sort:bool=True,          callback:BaseCallback=None,
                                     pin_memory:bool=False,      timeout:int=0,
                                     num_workers:int=0, worker_init_fn=None)
```

Bases: object

DataLoader that loads batches of data from a Dataset.

#### Parameters

- **dataset** – The Dataset object to load data from.
- **batch\_size** – Batch\_size. (default: 32)
- **device** – An instance of *torch.device* specifying which device the Variables are going to be created on.
- **stage** – One of “train”, “dev”, and “test”. (default: “train”)
- **resample** – Whether to resample data between epochs. only effective when *mode* of dataset is “pair”. (default: *True*)
- **shuffle** – Whether to shuffle data between epochs. (default: *False*)
- **sort** – Whether to sort data according to length\_right. (default: *True*)
- **callback** – BaseCallback. See *matchzoo.engine.base\_callback.BaseCallback* for more details.
- **pin\_memory** – If set to *True*, tensors will be copied into pinned memory. (default: *False*)
- **timeout** – The timeout value for collecting a batch from workers. ( default: 0)
- **num\_workers** – The number of subprocesses to use for data loading. 0 means that the data will be loaded in the main process. (default: 0)
- **worker\_init\_fn** – If not None, this will be called on each worker subprocess with the worker id (an int in [0, num\_workers - 1]) as input, after seeding and before data loading. (default: None)

#### Examples

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data(stage='train')
>>> preprocessor = mz.preprocessors.CDSSMPPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset = mz.dataloader.Dataset(data_processed, mode='point')
>>> padding_callback = mz.dataloader.callbacks.CDSSMPPadding()
>>> dataloader = mz.dataloader.DataLoader(
...     dataset, stage='train', callback=padding_callback)
>>> len(dataloader)
```

4

**id\_left**

*id\_left* getter.

**label**

*label* getter.

`__len__(self)`  
Get the total number of batches.

`init_epoch(self)`  
Resample, shuffle or sort the dataset for a new epoch.

`__iter__(self)`  
Iteration.

`_handle_callbacks_on_batch_unpacked(self, x, y)`

**class** `matchzoo.dataloader.DataLoaderBuilder` (*\*\*kwargs*)

Bases: `object`

DataLoader Bulider. In essense a wrapped partial function.

### Example

```
>>> import matchzoo as mz
>>> padding_callback = mz.dataloader.callbacks.CDSSMPadding()
>>> builder = mz.dataloader.DataLoaderBuilder(
...     stage='train', callback=padding_callback
... )
>>> data_pack = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.CDSSMPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset = mz.dataloader.Dataset(data_processed, mode='point')
>>> dataloder = builder.build(dataset)
>>> type(dataloder)
<class 'matchzoo.dataloader.dataloader.DataLoader'>
```

**build** (*self, dataset, \*\*kwargs*)

Build a DataLoader.

#### Parameters

- **dataset** – Dataset to build upon.
- **kwargs** – Additional keyword arguments to override the keyword arguments passed in `__init__`.

**class** `matchzoo.dataloader.DatasetBuilder` (*\*\*kwargs*)

Bases: `object`

Dataset Bulider. In essense a wrapped partial function.

### Example

```
>>> import matchzoo as mz
>>> builder = mz.dataloader.DatasetBuilder(
...     mode='point'
... )
>>> data = mz.datasets.toy.load_data()
>>> gen = builder.build(data)
>>> type(gen)
<class 'matchzoo.dataloader.dataset.Dataset'>
```

**build** (*self, data\_pack, \*\*kwargs*)

Build a Dataset.

### Parameters

- **data\_pack** – DataPack to build upon.
- **kwargs** – Additional keyword arguments to override the keyword arguments passed in `__init__`.

`matchzoo.datasets`

### Subpackages

`matchzoo.datasets.embeddings`

### Submodules

`matchzoo.datasets.embeddings.load_fasttext_embedding`

FastText embedding data loader.

### Module Contents

`matchzoo.datasets.embeddings.load_fasttext_embedding._fasttext_embedding_url = https://dl.fba`

`matchzoo.datasets.embeddings.load_fasttext_embedding.load_fasttext_embedding(language:str='en')`  
→  
`mz.embedding.Embedding`

Return the pretrained fasttext embedding.

**Parameters** **language** – the language of embedding. Supported language can be referred to “[https://github.com/facebookresearch/fastText/blob/master](https://github.com/facebookresearch/fastText/blob/master/docs/pretrained-vectors.md)” “/docs/pretrained-vectors.md”

**Returns** The `mz.embedding.Embedding` object.

`matchzoo.datasets.embeddings.load_glove_embedding`

GloVe Embedding data loader.

### Module Contents

`matchzoo.datasets.embeddings.load_glove_embedding._glove_embedding_url = http://nlp.stanford`

`matchzoo.datasets.embeddings.load_glove_embedding.load_glove_embedding(dimension:int=50)`  
→  
`mz.embedding.Embedding`

Return the pretrained glove embedding.

**Parameters** **dimension** – the size of embedding dimension, the value can only be 50, 100, or 300.

**Returns** The `mz.embedding.Embedding` object.

## Package Contents

`matchzoo.datasets.embeddings.load_glove_embedding` (*dimension:int=50*) → `mz.embedding.Embedding`

Return the pretrained glove embedding.

**Parameters** **dimension** – the size of embedding dimension, the value can only be 50, 100, or 300.

**Returns** The `mz.embedding.Embedding` object.

`matchzoo.datasets.embeddings.load_fasttext_embedding` (*language:str='en'*) → `mz.embedding.Embedding`

Return the pretrained fasttext embedding.

**Parameters** **language** – the language of embedding. Supported language can be referred to “[https://github.com/facebookresearch/fastText/blob/master](https://github.com/facebookresearch/fastText/blob/master/docs/pretrained-vectors.md)” “</docs/pretrained-vectors.md>”

**Returns** The `mz.embedding.Embedding` object.

`matchzoo.datasets.embeddings.DATA_ROOT`

`matchzoo.datasets.embeddings.EMBED_RANK`

`matchzoo.datasets.embeddings.EMBED_10`

`matchzoo.datasets.embeddings.EMBED_10_GLOVE`

`matchzoo.datasets.quora_qp`

## Submodules

`matchzoo.datasets.quora_qp.load_data`

Quora Question Pairs data loader.

## Module Contents

`matchzoo.datasets.quora_qp.load_data._url = https://firebasestorage.googleapis.com/v0/b/mt.`

`matchzoo.datasets.quora_qp.load_data.load_data` (*stage:str='train',*  
*task:str='classification',* *return\_classes:bool=False*) → *typing.Union[matchzoo.DataPack, tuple]*

Load QuoraQP data.

### Parameters

- **path** – *None* for download from quora, specific path for downloaded data.
- **stage** – One of *train*, *dev*, and *test*.
- **task** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance.
- **return\_classes** – Whether return classes for classification task.

**Returns** A `DataPack` if *ranking*, a tuple of (`DataPack`, classes) if *classification*.

`matchzoo.datasets.quora_qp.load_data._download_data()`

```
matchzoo.datasets.quora_qp.load_data._read_data(path, stage)
```

## Package Contents

```
matchzoo.datasets.quora_qp.load_data(stage:str='train',          task:str='classification',
                                     return_classes:bool=False)  →      typing.Union[matchzoo.DataPack, tuple]
```

Load QuoraQP data.

### Parameters

- **path** – *None* for download from quora, specific path for downloaded data.
- **stage** – One of *train*, *dev*, and *test*.
- **task** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance.
- **return\_classes** – Whether return classes for classification task.

**Returns** A `DataPack` if *ranking*, a tuple of (`DataPack`, classes) if *classification*.

```
matchzoo.datasets.snli
```

## Submodules

```
matchzoo.datasets.snli.load_data
```

SNLI data loader.

## Module Contents

```
matchzoo.datasets.snli.load_data._url = https://nlp.stanford.edu/projects/snli/snli_1.0.zip
matchzoo.datasets.snli.load_data.load_data(stage:str='train',    task:str='classification',
                                     target_label:str='entailment',      re-
                                     return_classes:bool=False)  →      typ-
                                     ing.Union[matchzoo.DataPack, tuple]
```

Load SNLI data.

### Parameters

- **stage** – One of *train*, *dev*, and *test*. (default: *train*)
- **task** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance. (default: *ranking*)
- **target\_label** – If *ranking*, chose one of *entailment*, *contradiction*, *neutral*, and - as the positive label. (default: *entailment*)
- **return\_classes** – *True* to return classes for classification task, *False* otherwise.

**Returns** A `DataPack` unless *task* is *classification* and *return\_classes* is *True*: a tuple of (`DataPack`, *classes*) in that case.

```
matchzoo.datasets.snli.load_data._download_data()
```

```
matchzoo.datasets.snli.load_data._read_data(path)
```

## Package Contents

```
matchzoo.datasets.snli.load_data(stage:str='train', task:str='classification', tar-
                                get_label:str='entailment', return_classes:bool=False)
                                → typing.Union[matchzoo.DataPack, tuple]
```

Load SNLI data.

### Parameters

- **stage** – One of *train*, *dev*, and *test*. (default: *train*)
- **task** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance. (default: *ranking*)
- **target\_label** – If *ranking*, chose one of *entailment*, *contradiction*, *neutral*, and *-* as the positive label. (default: *entailment*)
- **return\_classes** – *True* to return classes for classification task, *False* otherwise.

**Returns** A DataPack unless *task* is *classification* and *return\_classes* is *True*: a tuple of (*DataPack*, *classes*) in that case.

`matchzoo.datasets.toy`

## Package Contents

```
matchzoo.datasets.toy.load_data(stage:str='train', task:str='ranking', re-
                                turn_classes:bool=False) → typ-
                                ing.Union[matchzoo.DataPack,      typ-
                                ing.Tuple[matchzoo.DataPack, list]]
```

Load WikiQA data.

### Parameters

- **stage** – One of *train*, *dev*, and *test*.
- **task** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance.
- **return\_classes** – *True* to return classes for classification task, *False* otherwise.

**Returns** A DataPack unless *task* is *classification* and *return\_classes* is *True*: a tuple of (*DataPack*, *classes*) in that case.

## Example

```
>>> import matchzoo as mz
>>> stages = 'train', 'dev', 'test'
>>> tasks = 'ranking', 'classification'
>>> for stage in stages:
...     for task in tasks:
...         _ = mz.datasets.toy.load_data(stage, task)
```

`matchzoo.datasets.toy.load_embedding()`

`matchzoo.datasets.wiki_qa`

## Submodules

`matchzoo.datasets.wiki_qa.load_data`

WikiQA data loader.

## Module Contents

`matchzoo.datasets.wiki_qa.load_data._url = https://download.microsoft.com/download/E/5/F/E5F...`

```
matchzoo.datasets.wiki_qa.load_data.load_data(stage:str='train', task:str='ranking',
                                              filtered:bool=False,           re-
                                              turn_classes:bool=False) → typ-
                                              ing.Union[matchzoo.DataPack, tuple]
```

Load WikiQA data.

### Parameters

- **stage** – One of *train*, *dev*, and *test*.
- **task** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance.
- **filtered** – Whether remove the questions without correct answers.
- **return\_classes** – *True* to return classes for classification task, *False* otherwise.

**Returns** A `DataPack` unless *task* is *classification* and *return\_classes* is *True*: a tuple of (`DataPack`, *classes*) in that case.

`matchzoo.datasets.wiki_qa.load_data._download_data()`

`matchzoo.datasets.wiki_qa.load_data._read_data(path)`

## Package Contents

```
matchzoo.datasets.wiki_qa.load_data(stage:str='train', task:str='ranking', fil-
                                   tered:bool=False, return_classes:bool=False) →
                                   typing.Union[matchzoo.DataPack, tuple]
```

Load WikiQA data.

### Parameters

- **stage** – One of *train*, *dev*, and *test*.
- **task** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance.
- **filtered** – Whether remove the questions without correct answers.
- **return\_classes** – *True* to return classes for classification task, *False* otherwise.

**Returns** A `DataPack` unless *task* is *classification* and *return\_classes* is *True*: a tuple of (`DataPack`, *classes*) in that case.



## Package Contents

`matchzoo.datasets.list_available()`

`matchzoo.embedding`

## Submodules

`matchzoo.embedding.embedding`

Matchzoo toolkit for token embedding.

## Module Contents

**class** `matchzoo.embedding.embedding.Embedding` (*data:dict, output\_dim:int*)

Bases: `object`

Embedding class.

**Examples::**

```
>>> import matchzoo as mz
>>> train_raw = mz.datasets.toy.load_data()
>>> pp = mz.preprocessors.NaivePreprocessor()
>>> train = pp.fit_transform(train_raw, verbose=0)
>>> vocab_unit = mz.build_vocab_unit(train, verbose=0)
>>> term_index = vocab_unit.state['term_index']
>>> embed_path = mz.datasets.embeddings.EMBED_RANK
```

**To load from a file:**

```
>>> embedding = mz.embedding.load_from_file(embed_path)
>>> matrix = embedding.build_matrix(term_index)
>>> matrix.shape[0] == len(term_index)
True
```

**To build your own:**

```
>>> data = {'A':[0, 1], 'B':[2, 3]}
>>> embedding = mz.Embedding(data, 2)
>>> matrix = embedding.build_matrix({'A': 2, 'B': 1, '_PAD': 0})
>>> matrix.shape == (3, 2)
True
```

**build\_matrix** (*self, term\_index:typing.Union[dict, mz.preprocessors.units.Vocabulary.TermIndex], initializer=lambda: np.random.uniform(-0.2, 0.2))*

Build a matrix using *term\_index*.

### Parameters

- **term\_index** – A *dict* or *TermIndex* to build with.
- **initializer** – A callable that returns a default value for missing terms in data. (default: a random uniform distribution in range) *(-0.2, 0.2)*).

**Returns** A matrix.

`matchzoo.embedding.embedding.load_from_file` (*file\_path:str, mode:str='word2vec'*) → Embedding

Load embedding from *file\_path*.

#### Parameters

- **file\_path** – Path to file.
- **mode** – Embedding file format mode, one of 'word2vec', 'fasttext' or 'glove'. (default: 'word2vec')

**Returns** An `matchzoo.embedding.Embedding` instance.

## Package Contents

**class** `matchzoo.embedding.Embedding` (*data:dict, output\_dim:int*)

Bases: `object`

Embedding class.

#### Examples::

```
>>> import matchzoo as mz
>>> train_raw = mz.datasets.toy.load_data()
>>> pp = mz.preprocessors.NaivePreprocessor()
>>> train = pp.fit_transform(train_raw, verbose=0)
>>> vocab_unit = mz.build_vocab_unit(train, verbose=0)
>>> term_index = vocab_unit.state['term_index']
>>> embed_path = mz.datasets.embeddings.EMBED_RANK
```

#### To load from a file:

```
>>> embedding = mz.embedding.load_from_file(embed_path)
>>> matrix = embedding.build_matrix(term_index)
>>> matrix.shape[0] == len(term_index)
True
```

#### To build your own:

```
>>> data = {'A':[0, 1], 'B':[2, 3]}
>>> embedding = mz.Embedding(data, 2)
>>> matrix = embedding.build_matrix({'A': 2, 'B': 1, '_PAD': 0})
>>> matrix.shape == (3, 2)
True
```

**build\_matrix** (*self, term\_index:typing.Union[dict, mz.preprocessors.units.Vocabulary.TermIndex], initializer=lambda: np.random.uniform(-0.2, 0.2))*)

Build a matrix using *term\_index*.

#### Parameters

- **term\_index** – A *dict* or *TermIndex* to build with.
- **initializer** – A callable that returns a default value for missing terms in data. (default: a random uniform distribution in range) *(-0.2, 0.2)*).

**Returns** A matrix.

`matchzoo.embedding.load_from_file` (*file\_path:str, mode:str='word2vec'*) → Embedding

Load embedding from *file\_path*.

**Parameters**

- **file\_path** – Path to file.
- **mode** – Embedding file format mode, one of ‘word2vec’, ‘fasttext’ or ‘glove’.(default: ‘word2vec’)

**Returns** An *matchzoo.embedding.Embedding* instance.

`matchzoo.engine`

**Submodules**

`matchzoo.engine.base_callback`

Base callback.

**Module Contents**

**class** `matchzoo.engine.base_callback.BaseCallback`

Bases: `abc.ABC`

DataGenerator callback base class.

To build your own callbacks, inherit *mz.data\_generator.callbacks.Callback* and overrides corresponding methods.

A batch is processed in the following way:

- slice data pack based on batch index
- handle *on\_batch\_data\_pack* callbacks
- unpack data pack into x, y
- handle *on\_batch\_x\_y* callbacks
- return x, y

**on\_batch\_data\_pack** (*self*, *data\_pack*:*mz.DataPack*)  
*on\_batch\_data\_pack*.

**Parameters** **data\_pack** – a sliced DataPack before unpacking.

**on\_batch\_unpacked** (*self*, *x*:*dict*, *y*:*np.ndarray*)  
*on\_batch\_unpacked*.

**Parameters**

- **x** – unpacked x.
- **y** – unpacked y.

`matchzoo.engine.base_metric`

Metric base class and some related utilities.

## Module Contents

**class** matchzoo.engine.base\_metric.**BaseMetric**

Bases: abc.ABC

Metric base class.

**ALIAS = base\_metric**

**\_\_call\_\_** (*self*, *y\_true*:np.array, *y\_pred*:np.array)

Call to compute the metric.

**Parameters**

- **y\_true** – An array of ground truth labels.
- **y\_pred** – An array of predicted values.

**Returns** Evaluation of the metric.

**\_\_repr\_\_** (*self*)

**Returns** Formatted string representation of the metric.

**\_\_eq\_\_** (*self*, *other*)

**Returns** *True* if two metrics are equal, *False* otherwise.

**\_\_hash\_\_** (*self*)

**Returns** Hashing value using the metric as *str*.

**class** matchzoo.engine.base\_metric.**RankingMetric**

Bases: *matchzoo.engine.base\_metric.BaseMetric*

Ranking metric base class.

**ALIAS = ranking\_metric**

**class** matchzoo.engine.base\_metric.**ClassificationMetric**

Bases: *matchzoo.engine.base\_metric.BaseMetric*

Ranking metric base class.

**ALIAS = classification\_metric**

matchzoo.engine.base\_metric.**sort\_and\_couple** (*labels*:np.array, *scores*:np.array) → np.array

Zip the *labels* with *scores* into a single list.

**matchzoo.engine.base\_model**

Base Model.

## Module Contents

**class** matchzoo.engine.base\_model.**BaseModel** (*params*:typing.Optional[ParamTable]=None)

Bases: torch.nn.Module, abc.ABC

Abstract base class of all MatchZoo models.

MatchZoo models are wrapped over pytorch models. *params* is a set of model hyper-parameters that deterministically builds a model. In other words, *params*['model\_class'](params=params) of the same *params* always create models with the same structure.

**Parameters** *params* – Model hyper-parameters. (default: return value from *get\_default\_params()*)

### Example

```
>>> BaseModel() # doctest: +ELLIPSIS
Traceback (most recent call last):
...
TypeError: Can't instantiate abstract class BaseModel ...
>>> class MyModel(BaseModel):
...     def build(self):
...         pass
...     def forward(self):
...         pass
>>> isinstance(MyModel(), BaseModel)
True
```

### params

model parameters.

Type return

**classmethod** *get\_default\_params* (*cls*, *with\_embedding=False*, *with\_multi\_layer\_perceptron=False*)

Model default parameters.

The common usage is to instantiate `matchzoo.engine.ModelParams` first, then set the model specific parametrs.

### Examples

```
>>> class MyModel(BaseModel):
...     def build(self):
...         print(self._params['num_eggs'], 'eggs')
...         print('and', self._params['ham_type'])
...     def forward(self, greeting):
...         print(greeting)
...
...     @classmethod
...     def get_default_params(cls):
...         params = ParamTable()
...         params.add(Param('num_eggs', 512))
...         params.add(Param('ham_type', 'Parma Ham'))
...         return params
>>> my_model = MyModel()
>>> my_model.build()
512 eggs
and Parma Ham
>>> my_model('Hello MatchZoo!')
Hello MatchZoo!
```

Notice that all parameters must be serialisable for the entire model to be serialisable. Therefore, it's strongly recommended to use python native data types to store parameters.

**Returns** model parameters

**guess\_and\_fill\_missing\_params** (*self*, *verbose=1*)

Guess and fill missing parameters in *params*.

Use this method to automatically fill-in other hyper parameters. This involves some guessing so the parameter it fills could be wrong. For example, the default task is *Ranking*, and if we do not set it to *Classification* manually for data packs prepared for classification, then the shape of the model output and the data will mismatch.

**Parameters** *verbose* – Verbosity.

**\_set\_param\_default** (*self*, *name:str*, *default\_val:str*, *verbose:int=0*)

**classmethod get\_default\_preprocessor** (*cls*)

Model default preprocessor.

The preprocessor's transform should produce a correctly shaped data pack that can be used for training.

**Returns** Default preprocessor.

**classmethod get\_default\_padding\_callback** (*cls*)

Model default padding callback.

The padding callback's *on\_batch\_unpacked* would pad a batch of data to a fixed length.

**Returns** Default padding callback.

**build** (*self*)

Build model, each subclass need to implement this method.

**forward** (*self*, *\*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

**\_make\_embedding\_layer** (*self*, *num\_embeddings:int=0*, *embedding\_dim:int=0*, *freeze:bool=True*, *embedding:typing.Optional[np.ndarray]=None*, *\*\*kwargs*)

**Returns** an embedding module.

**\_make\_default\_embedding\_layer** (*self*, *\*\*kwargs*)

**Returns** an embedding module.

**\_make\_output\_layer** (*self*, *in\_features:int=0*, *activation:typing.Union[str, nn.Module]=None*)

**Returns** a correctly shaped torch module for model output.

**\_make\_perceptron\_layer** (*self*, *in\_features:int=0*, *out\_features:int=0*, *activation:nn.Module=nn.ReLU*)

**Returns** a perceptron layer.

**\_make\_multi\_layer\_perceptron\_layer** (*self*, *in\_features*)

**Returns** a multiple layer perceptron.

**matchzoo.engine.base\_preprocessor**

*BasePreprocessor* define input and output for processors.

## Module Contents

`matchzoo.engine.base_preprocessor.validate_context` (*func*)

Validate context in the preprocessor.

**class** `matchzoo.engine.base_preprocessor.BasePreprocessor`

*BasePreprocessor* to input handle data.

A preprocessor should be used in two steps. First, *fit*, then, *transform*. *fit* collects information into *context*, which includes everything the preprocessor needs to *transform* together with other useful information for later use. *fit* will only change the preprocessor's inner state but not the input data. In contrast, *transform* returns a modified copy of the input data without changing the preprocessor's inner state.

**DATA\_FILENAME** = `preprocessor.dill`

**context**

Return context.

**fit** (*self*, *data\_pack*: 'mz.DataPack', *verbose*:int=1)

Fit parameters on input data.

This method is an abstract base method, need to be implemented in the child class.

This method is expected to return itself as a callable object.

**Parameters**

- **data\_pack** – Datapack object to be fitted.
- **verbose** – Verbosity.

**transform** (*self*, *data\_pack*: 'mz.DataPack', *verbose*:int=1)

Transform input data to expected manner.

This method is an abstract base method, need to be implemented in the child class.

**Parameters**

- **data\_pack** – DataPack object to be transformed.
- **verbose** – Verbosity. or list of text-left, text-right tuples.

**fit\_transform** (*self*, *data\_pack*: 'mz.DataPack', *verbose*:int=1)

Call fit-transform.

**Parameters**

- **data\_pack** – DataPack object to be processed.
- **verbose** – Verbosity.

**save** (*self*, *dirpath*:typing.Union[str, Path])

Save the DSSMPreprocessor object.

A saved DSSMPreprocessor is represented as a directory with the *context* object (fitted parameters on training data), it will be saved by *pickle*.

**Parameters** *dirpath* – directory path of the saved DSSMPreprocessor.

**classmethod** `_default_units` (*cls*)

Prepare needed process units.

`matchzoo.engine.base_preprocessor.load_preprocessor` (*dirpath*:typing.Union[str, Path])

→ 'mz.DataPack'

Load the fitted *context*. The reverse function of `save()`.

**Parameters** `dirpath` – directory path of the saved model.

**Returns** a `DSSMPreprocessor` instance.

## `matchzoo.engine.base_task`

Base task.

## Module Contents

**class** `matchzoo.engine.base_task.BaseTask` (*losses=None, metrics=None*)

Bases: `abc.ABC`

Base Task, shouldn't be used directly.

**TYPE = base**

**losses**

Losses used in the task.

**Type** return

**metrics**

Metrics used in the task.

**Type** return

**output\_shape**

output shape of a single sample of the task.

**Type** return

**output\_dtype**

output data type for specific task.

**Type** return

**\_convert** (*self, identifiers, parse*)

**\_assure\_losses** (*self*)

**\_assure\_metrics** (*self*)

**classmethod** **list\_available\_losses** (*cls*)

**Returns** a list of available losses.

**classmethod** **list\_available\_metrics** (*cls*)

**Returns** a list of available metrics.

## `matchzoo.engine.hyper_spaces`

Hyper parameter search spaces wrapping *hyperopt*.



## Module Contents

**class** matchzoo.engine.hyper\_spaces.**HyperoptProxy** (*hyperopt\_func:typing.Callable[...  
hyperopt.pyll.Apply], \*\*kwargs*)

Bases: object

Hyperopt proxy class.

See *hyperopt*'s documentation for more details: <https://github.com/hyperopt/hyperopt/wiki/FMin>

Reason of these wrappers:

A hyper space in *hyperopt* requires a *label* to instantiate. This *label* is used later as a reference to original hyper space that is sampled. In *matchzoo*, hyper spaces are used in *matchzoo.engine.Param*. Only if a hyper space's label matches its parent *matchzoo.engine.Param*'s name, *matchzoo* can correctly back-referenced the parameter got sampled. This can be done by asking the user always use the same name for a parameter and its hyper space, but typos can occur. As a result, these wrappers are created to hide hyper spaces' *label*, and always correctly bind them with its parameter's name.

Examples::

```
>>> import matchzoo as mz
>>> from hyperopt.pyll.stochastic import sample
```

Basic Usage:

```
>>> model = mz.models.DenseBaseline()
>>> sample(model.params.hyper_space) # doctest: +SKIP
{'mlp_num_layers': 1.0, 'mlp_num_units': 274.0}
```

Arithmetic Operations:

```
>>> new_space = 2 ** mz.hyper_spaces.quniform(2, 6)
>>> model.params.get('mlp_num_layers').hyper_space = new_space
>>> sample(model.params.hyper_space) # doctest: +SKIP
{'mlp_num_layers': 8.0, 'mlp_num_units': 292.0}
```

**convert** (*self, name:str*)

Attach *name* as *hyperopt.hp*'s label.

**Parameters** *name* –

**Returns** a *hyperopt* ready search space

**\_\_add\_\_** (*self, other*)  
\_\_add\_\_.

**\_\_radd\_\_** (*self, other*)  
\_\_radd\_\_.

**\_\_sub\_\_** (*self, other*)  
\_\_sub\_\_.

**\_\_rsub\_\_** (*self, other*)  
\_\_rsub\_\_.

**\_\_mul\_\_** (*self, other*)  
\_\_mul\_\_.

```
__rmul__(self, other)
    __rmul__.

__truediv__(self, other)
    __truediv__.

__rtruediv__(self, other)
    __rtruediv__.

__floordiv__(self, other)
    __floordiv__.

__rfloordiv__(self, other)
    __rfloordiv__.

__pow__(self, other)
    __pow__.

__rpow__(self, other)
    __rpow__.

__neg__(self)
    __neg__.
```

```
matchzoo.engine.hyper_spaces._wrap_as_composite_func(self, other, func)
```

```
class matchzoo.engine.hyper_spaces.choice(options:list)
    Bases: matchzoo.engine.hyper_spaces.HyperoptProxy
    hyperopt.hp.choice() proxy.

    __str__(self)
```

**Returns** *str* representation of the hyper space.

```
class matchzoo.engine.hyper_spaces.quniform(low:numbers.Number,
                                             high:numbers.Number,
                                             q:numbers.Number=1)
    Bases: matchzoo.engine.hyper_spaces.HyperoptProxy
    hyperopt.hp.quniform() proxy.

    __str__(self)
```

**Returns** *str* representation of the hyper space.

```
class matchzoo.engine.hyper_spaces.uniform(low:numbers.Number,
                                           high:numbers.Number)
    Bases: matchzoo.engine.hyper_spaces.HyperoptProxy
    hyperopt.hp.uniform() proxy.

    __str__(self)
```

**Returns** *str* representation of the hyper space.

```
matchzoo.engine.hyper_spaces.sample(space)
    Take a sample in the hyper space.
```

This method is stateless, so the distribution of the samples is different from that of *tune* call. This function just gives a general idea of what a sample from the *space* looks like.

## Example

```
>>> import matchzoo as mz
>>> space = mz.models.DenseBaseline.get_default_params().hyper_space
>>> mz.hyper_spaces.sample(space) # doctest: +ELLIPSIS
{'mlp_num_fan_out': ...}
```

## matchzoo.engine.param

Parameter class.

## Module Contents

matchzoo.engine.param.SpaceType

```
class matchzoo.engine.param.Param(name:str, value:typing.Any=None, hy-
                                per_space:typing.Optional[SpaceType]=None, val-
                                idator:typing.Optional[typing.Callable[[typing.Any],
                                bool]]=None, desc:typing.Optional[str]=None)
```

Bases: object

Parameter class.

Basic usages with a name and value:

```
>>> param = Param('my_param', 10)
>>> param.name
'my_param'
>>> param.value
10
```

Use with a validator to make sure the parameter always keeps a valid value.

```
>>> param = Param(
...     name='my_param',
...     value=5,
...     validator=lambda x: 0 < x < 20
... )
>>> param.validator # doctest: +ELLIPSIS
<function <lambda> at 0x...>
>>> param.value
5
>>> param.value = 10
>>> param.value
10
>>> param.value = -1
Traceback (most recent call last):
...
ValueError: Validator not satisfied.
The validator's definition is as follows:
validator=lambda x: 0 < x < 20
```

Use with a hyper space. Setting up a hyper space for a parameter makes the parameter tunable in a matchzoo.engine.Tuner.

```
>>> from matchzoo.engine.hyper_spaces import quniform
>>> param = Param(
...     name='positive_num',
...     value=1,
...     hyper_space=quniform(low=1, high=5)
... )
>>> param.hyper_space # doctest: +ELLIPSIS
<matchzoo.engine.hyper_spaces.quniform object at ...>
>>> from hyperopt.pyll.stochastic import sample
>>> hyperopt_space = param.hyper_space.convert(param.name)
>>> samples = [sample(hyperopt_space) for _ in range(64)]
>>> set(samples) == {1, 2, 3, 4, 5}
True
```

The boolean value of a `Param` instance is only `True` when the value is not `None`. This is because some default falsy values like zero or an empty list are valid parameter values. In other words, the boolean value means to be “if the parameter value is filled”.

```
>>> param = Param('dropout')
>>> if param:
...     print('OK')
>>> param = Param('dropout', 0)
>>> if param:
...     print('OK')
OK
```

A `_pre_assignment_hook` is initialized as a data type convertor if the value is set as a number to keep data type consistency of the parameter. This conversion supports python built-in numbers, *numpy* numbers, and any number that inherits `numbers.Number`.

```
>>> param = Param('float_param', 0.5)
>>> param.value = 10
>>> param.value
10.0
>>> type(param.value)
<class 'float'>
```

**name**  
Name of the parameter.

**Type** return

**value**  
Value of the parameter.

**Type** return

**hyper\_space**  
Hyper space of the parameter.

**Type** return

**validator**  
Validator of the parameter.

**Type** return

**desc**  
Parameter description.

Type return

`_infer_pre_assignment_hook(self)`

`_validate(self, value)`

`__bool__(self)`

Returns *False* when the value is *None*, *True* otherwise.

`set_default(self, val, verbose=1)`

Set default value, has no effect if already has a value.

Parameters

- **val** – Default value to set.
- **verbose** – Verbosity.

`reset(self)`

Set the parameter's value to *None*, which means “not set”.

This method bypasses validator.

### Example

```
>>> import matchzoo as mz
>>> param = mz.Param(
...     name='str', validator=lambda x: isinstance(x, str))
>>> param.value = 'hello'
>>> param.value = None
Traceback (most recent call last):
...
ValueError: Validator not satisfied.
The validator's definition is as follows:
name='str', validator=lambda x: isinstance(x, str))
>>> param.reset()
>>> param.value is None
True
```

## matchzoo.engine.param\_table

Parameters table class.

### Module Contents

**class** matchzoo.engine.param\_table.ParamTable

Bases: object

Parameter table class.

### Example

```

>>> params = ParamTable()
>>> params.add(Param('ham', 'Parma Ham'))
>>> params.add(Param('egg', 'Over Easy'))
>>> params['ham']
'Parma Ham'
>>> params['egg']
'Over Easy'
>>> print(params)
ham                Parma Ham
egg                Over Easy
>>> params.add(Param('egg', 'Sunny side Up'))
Traceback (most recent call last):
...
ValueError: Parameter named egg already exists.
To re-assign parameter egg value, use `params["egg"] = value` instead.

```

**hyper\_space**

Hyper space of the table, a valid *hyperopt* graph.

**Type** return

**add** (*self*, *param:Param*)

**Parameters** *param* – parameter to add.

**get** (*self*, *key*)

**Returns** The parameter in the table named *key*.

**set** (*self*, *key*, *param:Param*)

Set *key* to parameter *param*.

**to\_frame** (*self*)

Convert the parameter table into a pandas data frame.

**Returns** A *pandas.DataFrame*.

**Example**

```

>>> import matchzoo as mz
>>> table = mz.ParamTable()
>>> table.add(mz.Param(name='x', value=10, desc='my x'))
>>> table.add(mz.Param(name='y', value=20, desc='my y'))
>>> table.to_frame()

```

	Name	Description	Value	Hyper-Space
0	x	my x	10	None
1	y	my y	20	None

**\_\_getitem\_\_** (*self*, *key:str*)

**Returns** The value of the parameter in the table named *key*.

**\_\_setitem\_\_** (*self*, *key:str*, *value:typing.Any*)

Set the value of the parameter named *key*.

**Parameters**

- **key** – Name of the parameter.
- **value** – New value of the parameter to set.

`__str__(self)`

**Returns** Pretty formatted parameter table.

`__iter__(self)`

**Returns** A iterator that iterates over all parameter instances.

`completed(self)`

**Returns** *True* if all params are filled, *False* otherwise.

### Example

```
>>> import matchzoo
>>> model = matchzoo.models.DenseBaseline()
>>> model.params.completed()
False
```

`keys(self)`

**Returns** Parameter table keys.

`__contains__(self, item)`

**Returns** *True* if parameter in parameters.

`update(self, other:dict)`

Update *self*.

Update *self* with the key/value pairs from *other*, overwriting existing keys. Notice that this does not add new keys to *self*.

This method is usually used by models to obtain useful information from a preprocessor's context.

**Parameters** *other* – The dictionary used update.

### Example

```
>>> import matchzoo as mz
>>> model = mz.models.DenseBaseline()
>>> prpr = model.get_default_preprocessor()
>>> _ = prpr.fit(mz.datasets.toy.load_data(), verbose=0)
>>> model.params.update(prpr.context)
```

## matchzoo.losses

### Submodules

#### matchzoo.losses.rank\_cross\_entropy\_loss

The rank cross entropy loss.

## Module Contents

**class** matchzoo.losses.rank\_cross\_entropy\_loss.**RankCrossEntropyLoss** (*num\_neg:int=1*)  
Bases: torch.nn.Module

Creates a criterion that measures rank cross entropy loss.

**\_\_constants\_\_** = ['num\_neg']

**num\_neg**  
*num\_neg* getter.

**forward** (*self*, *y\_pred:torch.Tensor*, *y\_true:torch.Tensor*)  
Calculate rank cross entropy loss.

### Parameters

- **y\_pred** – Predicted result.
- **y\_true** – Label.

**Returns** Rank cross loss.

**matchzoo.losses.rank\_hinge\_loss**

The rank hinge loss.

## Module Contents

**class** matchzoo.losses.rank\_hinge\_loss.**RankHingeLoss** (*num\_neg:int=1*, *margin:float=1.0*, *reduction:str='mean'*)  
Bases: torch.nn.Module

Creates a criterion that measures rank hinge loss.

Given inputs  $x_1$ ,  $x_2$ , two 1D mini-batch *Tensors*, and a label 1D mini-batch tensor  $y$  (containing 1 or -1).

If  $y = 1$  then it assumed the first input should be ranked higher (have a larger value) than the second input, and vice-versa for  $y = -1$ .

The loss function for each sample in the mini-batch is:

$$loss_{x,y} = \max(0, -y * (x_1 - x_2) + margin)$$

**\_\_constants\_\_** = ['num\_neg', 'margin', 'reduction']

**num\_neg**  
*num\_neg* getter.

**margin**  
*margin* getter.

**forward** (*self*, *y\_pred:torch.Tensor*, *y\_true:torch.Tensor*)  
Calculate rank hinge loss.

### Parameters

- **y\_pred** – Predicted result.



- **y\_true** – Label.

**Returns** Hinge loss computed by user-defined margin.

## Package Contents

**class** matchzoo.losses.**RankCrossEntropyLoss** (*num\_neg:int=1*)

Bases: torch.nn.Module

Creates a criterion that measures rank cross entropy loss.

**\_\_constants\_\_** = ['num\_neg']

**num\_neg**

*num\_neg* getter.

**forward** (*self, y\_pred:torch.Tensor, y\_true:torch.Tensor*)

Calculate rank cross entropy loss.

### Parameters

- **y\_pred** – Predicted result.
- **y\_true** – Label.

**Returns** Rank cross loss.

**class** matchzoo.losses.**RankHingeLoss** (*num\_neg:int=1, margin:float=1.0, reduction:str='mean'*)

Bases: torch.nn.Module

Creates a criterion that measures rank hinge loss.

Given inputs  $x_1$ ,  $x_2$ , two 1D mini-batch *Tensors*, and a label 1D mini-batch tensor  $y$  (containing 1 or -1).

If  $y = 1$  then it assumed the first input should be ranked higher (have a larger value) than the second input, and vice-versa for  $y = -1$ .

The loss function for each sample in the mini-batch is:

$$loss_{x,y} = \max(0, -y * (x_1 - x_2) + margin)$$

**\_\_constants\_\_** = ['num\_neg', 'margin', 'reduction']

**num\_neg**

*num\_neg* getter.

**margin**

*margin* getter.

**forward** (*self, y\_pred:torch.Tensor, y\_true:torch.Tensor*)

Calculate rank hinge loss.

### Parameters

- **y\_pred** – Predicted result.
- **y\_true** – Label.

**Returns** Hinge loss computed by user-defined margin.

`matchzoo.metrics`

## Submodules

`matchzoo.metrics.accuracy`

Accuracy metric for Classification.

## Module Contents

```
class matchzoo.metrics.accuracy.Accuracy
    Bases: matchzoo.engine.base_metric.ClassificationMetric
    Accuracy metric.

    ALIAS = ['accuracy', 'acc']

    __repr__(self)
        Returns Formated string representation of the metric.

    __call__(self, y_true:np.array, y_pred:np.array)
        Calculate accuracy.
```

## Example

```
>>> import numpy as np
>>> y_true = np.array([1])
>>> y_pred = np.array([[0, 1]])
>>> Accuracy()(y_true, y_pred)
1.0
```

### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Accuracy.

`matchzoo.metrics.average_precision`

Average precision metric for ranking.

## Module Contents

```
class matchzoo.metrics.average_precision.AveragePrecision (threshold:float=0.0)
    Bases: matchzoo.engine.base_metric.RankingMetric
    Average precision metric.

    ALIAS = ['average_precision', 'ap']

    __repr__(self)
```

**Returns** Formatted string representation of the metric.

**\_\_call\_\_** (*self*, *y\_true*:*np.array*, *y\_pred*:*np.array*)  
Calculate average precision (area under PR curve).

### Example

```
>>> y_true = [0, 1]
>>> y_pred = [0.1, 0.6]
>>> round(AveragePrecision()(y_true, y_pred), 2)
0.75
>>> round(AveragePrecision()([], []), 2)
0.0
```

### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Average precision.

## matchzoo.metrics.cross\_entropy

CrossEntropy metric for Classification.

### Module Contents

```
class matchzoo.metrics.cross_entropy.CrossEntropy
    Bases: matchzoo.engine.base_metric.ClassificationMetric
    Cross entropy metric.
    ALIAS = ['cross_entropy', 'ce']
    __repr__ (self)
        Returns Formatted string representation of the metric.
    __call__ (self, y_true:np.array, y_pred:np.array, eps:float=1e-12)
        Calculate cross entropy.
```

### Example

```
>>> y_true = [0, 1]
>>> y_pred = [[0.25, 0.25], [0.01, 0.90]]
>>> CrossEntropy()(y_true, y_pred)
0.7458274358333028
```

### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

- **eps** – The Log loss is undefined for  $p=0$  or  $p=1$ , so probabilities are clipped to  $\max(\text{eps}, \min(1 - \text{eps}, p))$ .

**Returns** Average precision.

#### `matchzoo.metrics.discounted_cumulative_gain`

Discounted cumulative gain metric for ranking.

### Module Contents

```
class matchzoo.metrics.discounted_cumulative_gain.DiscountedCumulativeGain(k:int=1,  
                                                                           thresh-  
                                                                           old:float=0.0)
```

Bases: `matchzoo.engine.base_metric.RankingMetric`

Disconunted cumulative gain metric.

**ALIAS** = ['discounted\_cumulative\_gain', 'dcg']

`__repr__`(*self*)

**Returns** Formated string representation of the metric.

`__call__`(*self, y\_true:np.array, y\_pred:np.array*)

Calculate discounted cumulative gain (dcg).

Relevance is positive real values or binary values.

### Example

```
>>> y_true = [0, 1, 2, 0]
>>> y_pred = [0.4, 0.2, 0.5, 0.7]
>>> DiscountedCumulativeGain(1)(y_true, y_pred)
0.0
>>> round(DiscountedCumulativeGain(k=-1)(y_true, y_pred), 2)
0.0
>>> round(DiscountedCumulativeGain(k=2)(y_true, y_pred), 2)
2.73
>>> round(DiscountedCumulativeGain(k=3)(y_true, y_pred), 2)
2.73
>>> type(DiscountedCumulativeGain(k=1)(y_true, y_pred))
<class 'float'>
```

### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Discounted cumulative gain.

#### `matchzoo.metrics.mean_average_precision`

Mean average precision metric for ranking.

## Module Contents

**class** matchzoo.metrics.mean\_average\_precision.**MeanAveragePrecision** (*threshold:float=0.0*)

Bases: *matchzoo.engine.base\_metric.RankingMetric*

Mean average precision metric.

**ALIAS** = ['mean\_average\_precision', 'map']

**\_\_repr\_\_** (*self*)

**Returns** Formated string representation of the metric.

**\_\_call\_\_** (*self, y\_true:np.array, y\_pred:np.array*)

Calculate mean average precision.

### Example

```
>>> y_true = [0, 1, 0, 0]
>>> y_pred = [0.1, 0.6, 0.2, 0.3]
>>> MeanAveragePrecision()(y_true, y_pred)
1.0
```

### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Mean average precision.

**matchzoo.metrics.mean\_reciprocal\_rank**

Mean reciprocal ranking metric.

## Module Contents

**class** matchzoo.metrics.mean\_reciprocal\_rank.**MeanReciprocalRank** (*threshold:float=0.0*)

Bases: *matchzoo.engine.base\_metric.RankingMetric*

Mean reciprocal rank metric.

**ALIAS** = ['mean\_reciprocal\_rank', 'mrr']

**\_\_repr\_\_** (*self*)

**Returns** Formated string representation of the metric.

**\_\_call\_\_** (*self, y\_true:np.array, y\_pred:np.array*)

Calculate reciprocal of the rank of the first relevant item.

### Example

```
>>> import numpy as np
>>> y_pred = np.asarray([0.2, 0.3, 0.7, 1.0])
>>> y_true = np.asarray([1, 0, 0, 0])
>>> MeanReciprocalRank()(y_true, y_pred)
0.25
```

#### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Mean reciprocal rank.

### `matchzoo.metrics.normalized_discounted_cumulative_gain`

Normalized discounted cumulative gain metric for ranking.

#### Module Contents

**class** `matchzoo.metrics.normalized_discounted_cumulative_gain.NormalizedDiscountedCumulativeGain`

Bases: `matchzoo.engine.base_metric.RankingMetric`

Normalized discounted cumulative gain metric.

**ALIAS** = ['normalized\_discounted\_cumulative\_gain', 'ndcg']

**\_\_repr\_\_** (*self*)

**Returns** Formated string representation of the metric.

**\_\_call\_\_** (*self*, *y\_true:np.array*, *y\_pred:np.array*)

Calculate normalized discounted cumulative gain (ndcg).

Relevance is positive real values or binary values.

#### Example

```
>>> y_true = [0, 1, 2, 0]
>>> y_pred = [0.4, 0.2, 0.5, 0.7]
>>> ndcg = NormalizedDiscountedCumulativeGain
>>> ndcg(k=1)(y_true, y_pred)
0.0
>>> round(ndcg(k=2)(y_true, y_pred), 2)
0.52
>>> round(ndcg(k=3)(y_true, y_pred), 2)
0.52
>>> type(ndcg()(y_true, y_pred))
<class 'float'>
```

#### Parameters

- **y\_true** – The ground true label of each document.

- **y\_pred** – The predicted scores of each document.

**Returns** Normalized discounted cumulative gain.

## matchzoo.metrics.precision

Precision for ranking.

## Module Contents

**class** matchzoo.metrics.precision.**Precision** (*k:int=1, threshold:float=0.0*)

Bases: *matchzoo.engine.base\_metric.RankingMetric*

Precision metric.

**ALIAS** = **precision**

**\_\_repr\_\_** (*self*)

**Returns** Formated string representation of the metric.

**\_\_call\_\_** (*self, y\_true:np.array, y\_pred:np.array*)

Calculate precision@k.

## Example

```
>>> y_true = [0, 0, 0, 1]
>>> y_pred = [0.2, 0.4, 0.3, 0.1]
>>> Precision(k=1)(y_true, y_pred)
0.0
>>> Precision(k=2)(y_true, y_pred)
0.0
>>> Precision(k=4)(y_true, y_pred)
0.25
>>> Precision(k=5)(y_true, y_pred)
0.2
```

## Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Precision @ k

**Raises** ValueError: len(r) must be >= k.

## Package Contents

**class** matchzoo.metrics.**Precision** (*k:int=1, threshold:float=0.0*)

Bases: *matchzoo.engine.base\_metric.RankingMetric*

Precision metric.

**ALIAS** = **precision**

**\_\_repr\_\_** (*self*)

**Returns** Formatted string representation of the metric.

**\_\_call\_\_** (*self*, *y\_true*:*np.array*, *y\_pred*:*np.array*)  
Calculate precision@k.

### Example

```
>>> y_true = [0, 0, 0, 1]
>>> y_pred = [0.2, 0.4, 0.3, 0.1]
>>> Precision(k=1) (y_true, y_pred)
0.0
>>> Precision(k=2) (y_true, y_pred)
0.0
>>> Precision(k=4) (y_true, y_pred)
0.25
>>> Precision(k=5) (y_true, y_pred)
0.2
```

### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Precision @ k

**Raises** ValueError: len(r) must be >= k.

**class** matchzoo.metrics.**DiscountedCumulativeGain** (*k:int=1*, *threshold:float=0.0*)

Bases: *matchzoo.engine.base\_metric.RankingMetric*

Disconuted cumulative gain metric.

**ALIAS** = ['discounted\_cumulative\_gain', 'dcg']

**\_\_repr\_\_** (*self*)

**Returns** Formatted string representation of the metric.

**\_\_call\_\_** (*self*, *y\_true*:*np.array*, *y\_pred*:*np.array*)  
Calculate discounted cumulative gain (dcg).

Relevance is positive real values or binary values.

### Example

```
>>> y_true = [0, 1, 2, 0]
>>> y_pred = [0.4, 0.2, 0.5, 0.7]
>>> DiscountedCumulativeGain(1) (y_true, y_pred)
0.0
>>> round(DiscountedCumulativeGain(k=-1) (y_true, y_pred), 2)
0.0
>>> round(DiscountedCumulativeGain(k=2) (y_true, y_pred), 2)
2.73
>>> round(DiscountedCumulativeGain(k=3) (y_true, y_pred), 2)
2.73
>>> type(DiscountedCumulativeGain(k=1) (y_true, y_pred))
<class 'float'>
```



**Parameters**

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Discounted cumulative gain.

**class** matchzoo.metrics.**MeanReciprocalRank** (*threshold:float=0.0*)

Bases: *matchzoo.engine.base\_metric.RankingMetric*

Mean reciprocal rank metric.

**ALIAS** = ['mean\_reciprocal\_rank', 'mrr']

**\_\_repr\_\_** (*self*)

**Returns** Formated string representation of the metric.

**\_\_call\_\_** (*self, y\_true:np.array, y\_pred:np.array*)

Calculate reciprocal of the rank of the first relevant item.

**Example**

```
>>> import numpy as np
>>> y_pred = np.asarray([0.2, 0.3, 0.7, 1.0])
>>> y_true = np.asarray([1, 0, 0, 0])
>>> MeanReciprocalRank()(y_true, y_pred)
0.25
```

**Parameters**

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Mean reciprocal rank.

**class** matchzoo.metrics.**MeanAveragePrecision** (*threshold:float=0.0*)

Bases: *matchzoo.engine.base\_metric.RankingMetric*

Mean average precision metric.

**ALIAS** = ['mean\_average\_precision', 'map']

**\_\_repr\_\_** (*self*)

**Returns** Formated string representation of the metric.

**\_\_call\_\_** (*self, y\_true:np.array, y\_pred:np.array*)

Calculate mean average precision.

**Example**

```
>>> y_true = [0, 1, 0, 0]
>>> y_pred = [0.1, 0.6, 0.2, 0.3]
>>> MeanAveragePrecision()(y_true, y_pred)
1.0
```

**Parameters**

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Mean average precision.

```
class matchzoo.metrics.NormalizedDiscountedCumulativeGain (k:int=1,      thresh-  
                                                         old:float=0.0)
```

Bases: *matchzoo.engine.base\_metric.RankingMetric*

Normalized discounted cumulative gain metric.

**ALIAS** = ['normalized\_discounted\_cumulative\_gain', 'ndcg']

**\_\_repr\_\_** (self)

**Returns** Formated string representation of the metric.

**\_\_call\_\_** (self, y\_true:np.array, y\_pred:np.array)

Calculate normalized discounted cumulative gain (ndcg).

Relevance is positive real values or binary values.

### Example

```
>>> y_true = [0, 1, 2, 0]
>>> y_pred = [0.4, 0.2, 0.5, 0.7]
>>> ndcg = NormalizedDiscountedCumulativeGain
>>> ndcg(k=1) (y_true, y_pred)
0.0
>>> round(ndcg(k=2) (y_true, y_pred), 2)
0.52
>>> round(ndcg(k=3) (y_true, y_pred), 2)
0.52
>>> type(ndcg() (y_true, y_pred))
<class 'float'>
```

### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Normalized discounted cumulative gain.

```
class matchzoo.metrics.Accuracy
```

Bases: *matchzoo.engine.base\_metric.ClassificationMetric*

Accuracy metric.

**ALIAS** = ['accuracy', 'acc']

**\_\_repr\_\_** (self)

**Returns** Formated string representation of the metric.

**\_\_call\_\_** (self, y\_true:np.array, y\_pred:np.array)

Calculate accuracy.

### Example

```
>>> import numpy as np
>>> y_true = np.array([1])
>>> y_pred = np.array([[0, 1]])
>>> Accuracy()(y_true, y_pred)
1.0
```

#### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.

**Returns** Accuracy.

**class** matchzoo.metrics.**CrossEntropy**

Bases: *matchzoo.engine.base\_metric.ClassificationMetric*

Cross entropy metric.

**ALIAS** = ['cross\_entropy', 'ce']

**\_\_repr\_\_** (*self*)

**Returns** Formated string representation of the metric.

**\_\_call\_\_** (*self*, *y\_true:np.array*, *y\_pred:np.array*, *eps:float=1e-12*)  
Calculate cross entropy.

### Example

```
>>> y_true = [0, 1]
>>> y_pred = [[0.25, 0.25], [0.01, 0.90]]
>>> CrossEntropy()(y_true, y_pred)
0.7458274358333028
```

#### Parameters

- **y\_true** – The ground true label of each document.
- **y\_pred** – The predicted scores of each document.
- **eps** – The Log loss is undefined for p=0 or p=1, so probabilities are clipped to max(eps, min(1 - eps, p)).

**Returns** Average precision.

`matchzoo.metrics.list_available()` → list

**matchzoo.models**

### Submodules

**matchzoo.models.arci**

An implementation of ArcI Model.

## Module Contents

**class** matchzoo.models.arci.**ArcI**

Bases: *matchzoo.engine.base\_model.BaseModel*

ArcI Model.

### Examples

```
>>> model = ArcI()
>>> model.params['left_filters'] = [32]
>>> model.params['right_filters'] = [32]
>>> model.params['left_kernel_sizes'] = [3]
>>> model.params['right_kernel_sizes'] = [3]
>>> model.params['left_pool_sizes'] = [2]
>>> model.params['right_pool_sizes'] = [4]
>>> model.params['conv_activation_func'] = 'relu'
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 64
>>> model.params['mlp_num_fan_out'] = 32
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.params['dropout_rate'] = 0.5
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** **get\_default\_params** (*cls*)

**Returns** model default parameters.

**classmethod** **get\_default\_padding\_callback** (*cls*)

**Returns** Default padding callback.

**build** (*self*)

Build model structure.

ArcI use Siamese arthitecture.

**forward** (*self, inputs*)

Forward.

**classmethod** **\_make\_conv\_pool\_block** (*cls, in\_channels:int, out\_channels:int, kernel\_size:int, activation:nn.Module, pool\_size:int*)

Make conv pool block.

**matchzoo.models.arcii**

An implementation of ArcII Model.

## Module Contents

**class** matchzoo.models.arcii.**ArcII**

Bases: *matchzoo.engine.base\_model.BaseModel*

ArcII Model.

```

Examples: >>> model = ArcII() >>> model.params['embedding_output_dim'] = 300
>>> model.params['kernel_1d_count'] = 32 >>> model.params['kernel_1d_size'] = 3 >>>
model.params['kernel_2d_count'] = [16, 32] >>> model.params['kernel_2d_size'] = [[3, 3], [3, 3]] >>>
model.params['pool_2d_size'] = [[2, 2], [2, 2]] >>> model.guess_and_fill_missing_params(verbose=0) >>>
model.build()

```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build**(*self*)

Build model structure.

ArcII has the desirable property of letting two sentences meet before their own high-level representations mature.

**forward**(*self*, *inputs*)

Forward.

**classmethod** `_make_conv_pool_block(cls, in_channels:int, out_channels:int, kernel_size:tuple, activation:nn.Module, pool_size:tuple)`

Make conv pool block.

## `matchzoo.models.bert`

An implementation of Bert Model.

### Module Contents

**class** `matchzoo.models.bert.Bert`

Bases: `matchzoo.engine.base_model.BaseModel`

Bert Model.

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**build**(*self*)

Build model structure.

**forward**(*self*, *inputs*)

Forward.

## `matchzoo.models.bimpm`

An implementation of BiMPM Model.

### Module Contents

**class** `matchzoo.models.bimpm.BiMPM`

Bases: `matchzoo.engine.base_model.BaseModel`

BiMPM Model.

Reference: - <https://github.com/galsang/BIMPM-pytorch/blob/master/model/BIMPM.py>

## Examples

```
>>> model = BiMPM()
>>> model.params['num_perspective'] = 4
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build**(*self*)

Make function layers.

**forward**(*self*, *inputs*)

Forward.

**reset\_parameters**(*self*)

Init Parameters.

**dropout**(*self*, *v*)

Dropout Layer.

`matchzoo.models.bimpmp.mp_matching_func(v1, v2, w)`

Basic mp\_matching\_func.

**Parameters**

- **v1** – (batch, seq\_len, hidden\_size)
- **v2** – (batch, seq\_len, hidden\_size) or (batch, hidden\_size)
- **w** – (num\_psp, hidden\_size)

**Returns** (batch, num\_psp)

`matchzoo.models.bimpmp.mp_matching_func_pairwise(v1, v2, w)`

Basic mp\_matching\_func\_pairwise.

**Parameters**

- **v1** – (batch, seq\_len1, hidden\_size)
- **v2** – (batch, seq\_len2, hidden\_size)
- **w** – (num\_psp, hidden\_size)

:param num\_psp :return: (batch, num\_psp, seq\_len1, seq\_len2)

`matchzoo.models.bimpmp.attention(v1, v2)`

Attention.

**Parameters**

- **v1** – (batch, seq\_len1, hidden\_size)
- **v2** – (batch, seq\_len2, hidden\_size)

**Returns** (batch, seq\_len1, seq\_len2)

`matchzoo.models.bimpm.div_with_small_value(n, d, eps=1e-08)`  
 Small values are replaced by 1e-8 to prevent it from exploding.

**Parameters**

- **n** – tensor
- **d** – tensor

**Returns** n/d: tensor

`matchzoo.models.cdssm`

An implementation of CDSSM (CLSM) model.

## Module Contents

**class** `matchzoo.models.cdssm.CDSSM`

Bases: `matchzoo.engine.base_model.BaseModel`

CDSSM Model implementation.

Learning Semantic Representations Using Convolutional Neural Networks for Web Search. (2014a) A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. (2014b)

## Examples

```
>>> import matchzoo as mz
>>> model = CDSSM()
>>> model.params['task'] = mz.tasks.Ranking()
>>> model.params['vocab_size'] = 4
>>> model.params['filters'] = 32
>>> model.params['kernel_size'] = 3
>>> model.params['conv_activation_func'] = 'relu'
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_preprocessor(cls)`

**Returns** Default preprocessor.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**\_create\_base\_network(self)**

Apply conv and maxpooling operation towards to each letter-ngram.

The input shape is *fixed\_text\_length* \* *number of letter-ngram*, as described in the paper, *n* is 3, *number of letter-trigram* is about 30,000 according to their observation.

**Returns** A `nn.Module` of CDSSM network, tensor in tensor out.

**build**(*self*)

Build model structure.

CDSSM use Siamese architecture.

**forward**(*self*, *inputs*)

Forward.

**guess\_and\_fill\_missing\_params**(*self*, *verbose*:int=1)

Guess and fill missing parameters in params.

Use this method to automatically fill-in hyper parameters. This involves some guessing so the parameter it fills could be wrong. For example, the default task is *Ranking*, and if we do not set it to *Classification* manually for data packs prepared for classification, then the shape of the model output and the data will mismatch.

**Parameters** *verbose* – Verbosity.

**class** matchzoo.models.cdssm.**Squeeze**

Bases: torch.nn.Module

Squeeze.

**forward**(*self*, *x*)

Forward.

**matchzoo.models.conv\_knrm**

An implementation of ConvKNRM Model.

## Module Contents

**class** matchzoo.models.conv\_knrm.**ConvKNRM**

Bases: *matchzoo.engine.base\_model.BaseModel*

ConvKNRM Model.

## Examples

```
>>> model = ConvKNRM()
>>> model.params['filters'] = 128
>>> model.params['conv_activation_func'] = 'tanh'
>>> model.params['max_ngram'] = 3
>>> model.params['use_crossmatch'] = True
>>> model.params['kernel_num'] = 11
>>> model.params['sigma'] = 0.1
>>> model.params['exact_sigma'] = 0.001
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** **get\_default\_params**(*cls*)

**Returns** model default parameters.

**build**(*self*)

Build model structure.



**forward** (*self*, *inputs*)  
Forward.

## `matchzoo.models.dense_baseline`

A simple densely connected baseline model.

### Module Contents

**class** `matchzoo.models.dense_baseline.DenseBaseline`

Bases: `matchzoo.engine.base_model.BaseModel`

A simple densely connected baseline model.

### Examples

```
>>> model = DenseBaseline()
>>> model.params['mlp_num_layers'] = 2
>>> model.params['mlp_num_units'] = 300
>>> model.params['mlp_num_fan_out'] = 128
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*)

**Returns** model default parameters.

**build** (*self*)  
Build.

**forward** (*self*, *inputs*)  
Forward.

## `matchzoo.models.drmm`

An implementation of DRMM Model.

### Module Contents

**class** `matchzoo.models.drmm.DRMM`

Bases: `matchzoo.engine.base_model.BaseModel`

DRMM Model.

### Examples

```
>>> model = DRMM()
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 5
>>> model.params['mlp_num_fan_out'] = 1
>>> model.params['mlp_activation_func'] = 'tanh'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build**(*self*)

Build model structure.

**forward**(*self*, *inputs*)

Forward.

`matchzoo.models.drmmtks`

An implementation of DRMMTKS Model.

## Module Contents

**class** `matchzoo.models.drmmtks.DRMMTKS`

Bases: `matchzoo.engine.base_model.BaseModel`

DRMMTKS Model.

## Examples

```
>>> model = DRMMTKS()
>>> model.params['top_k'] = 10
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 5
>>> model.params['mlp_num_fan_out'] = 1
>>> model.params['mlp_activation_func'] = 'tanh'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**build**(*self*)

Build model structure.

**forward**(*self*, *inputs*)

Forward.

## matchzoo.models.dssm

An implementation of DSSM, Deep Structured Semantic Model.

### Module Contents

**class** matchzoo.models.dssm.DSSM  
 Bases: *matchzoo.engine.base\_model.BaseModel*  
 Deep structured semantic model.

### Examples

```
>>> model = DSSM()
>>> model.params['mlp_num_layers'] = 3
>>> model.params['mlp_num_units'] = 300
>>> model.params['mlp_num_fan_out'] = 128
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** get\_default\_params(*cls*)

Returns model default parameters.

**classmethod** get\_default\_preprocessor(*cls*)

Returns Default preprocessor.

**classmethod** get\_default\_padding\_callback(*cls*)

Returns Default padding callback.

**build**(*self*)

Build model structure.

DSSM use Siamese architecture.

**forward**(*self, inputs*)

Forward.

## matchzoo.models.esim

An implementation of ESIM Model.

### Module Contents

**class** matchzoo.models.esim.ESIM  
 Bases: *matchzoo.engine.base\_model.BaseModel*  
 ESIM Model.

## Examples

```
>>> model = ESIM()
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build**(*self*)

Instantiating layers.

**forward**(*self*, *inputs*)

Forward.

### `matchzoo.models.knrm`

An implementation of KNRM Model.

## Module Contents

**class** `matchzoo.models.knrm.KNRM`

Bases: `matchzoo.engine.base_model.BaseModel`

KNRM Model.

## Examples

```
>>> model = KNRM()
>>> model.params['kernel_num'] = 11
>>> model.params['sigma'] = 0.1
>>> model.params['exact_sigma'] = 0.001
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**build**(*self*)

Build model structure.

**forward**(*self*, *inputs*)

Forward.

### `matchzoo.models.matchlstm`

An implementation of Match LSTM Model.

## Module Contents

**class** matchzoo.models.matchlstm.**MatchLSTM**

Bases: *matchzoo.engine.base\_model.BaseModel*

MatchLSTM Model.

<https://github.com/shuohangwang/mprc/blob/master/qa/rankerReader.lua>.

## Examples

```
>>> model = MatchLSTM()
>>> model.params['dropout'] = 0.2
>>> model.params['hidden_size'] = 200
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build**(*self*)

Instantiating layers.

**forward**(*self*, *inputs*)

Forward.

## Package Contents

**class** matchzoo.models.**DenseBaseline**

Bases: *matchzoo.engine.base\_model.BaseModel*

A simple densely connected baseline model.

## Examples

```
>>> model = DenseBaseline()
>>> model.params['mlp_num_layers'] = 2
>>> model.params['mlp_num_units'] = 300
>>> model.params['mlp_num_fan_out'] = 128
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**build**(*self*)

Build.

**forward**(*self*, *inputs*)

Forward.

**class** matchzoo.models.DSSM

Bases: *matchzoo.engine.base\_model.BaseModel*

Deep structured semantic model.

### Examples

```
>>> model = DSSM()
>>> model.params['mlp_num_layers'] = 3
>>> model.params['mlp_num_units'] = 300
>>> model.params['mlp_num_fan_out'] = 128
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** get\_default\_params(*cls*)

**Returns** model default parameters.

**classmethod** get\_default\_preprocessor(*cls*)

**Returns** Default preprocessor.

**classmethod** get\_default\_padding\_callback(*cls*)

**Returns** Default padding callback.

**build**(*self*)

Build model structure.

DSSM use Siamese architecture.

**forward**(*self, inputs*)

Forward.

**class** matchzoo.models.CDSSM

Bases: *matchzoo.engine.base\_model.BaseModel*

CDSSM Model implementation.

Learning Semantic Representations Using Convolutional Neural Networks for Web Search. (2014a) A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. (2014b)

### Examples

```
>>> import matchzoo as mz
>>> model = CDSSM()
>>> model.params['task'] = mz.tasks.Ranking()
>>> model.params['vocab_size'] = 4
>>> model.params['filters'] = 32
>>> model.params['kernel_size'] = 3
>>> model.params['conv_activation_func'] = 'relu'
>>> model.build()
```

**classmethod** get\_default\_params(*cls*)

**Returns** model default parameters.

**classmethod** get\_default\_preprocessor(*cls*)

**Returns** Default preprocessor.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**\_\_create\_base\_network**(*self*)

Apply conv and maxpooling operation towards to each letter-ngram.

The input shape is *fixed\_text\_length* \* *number of letter-ngram*, as described in the paper, *n* is 3, *number of letter-trigram* is about 30,000 according to their observation.

**Returns** A `nn.Module` of CDSSM network, tensor in tensor out.

**build**(*self*)

Build model structure.

CDSSM use Siamese architecture.

**forward**(*self*, *inputs*)

Forward.

**guess\_and\_fill\_missing\_params**(*self*, *verbose*:int=1)

Guess and fill missing parameters in *params*.

Use this method to automatically fill-in hyper parameters. This involves some guessing so the parameter it fills could be wrong. For example, the default task is *Ranking*, and if we do not set it to *Classification* manually for data packs prepared for classification, then the shape of the model output and the data will mismatch.

**Parameters** *verbose* – Verbosity.

**class** `matchzoo.models.DRMM`

Bases: `matchzoo.engine.base_model.BaseModel`

DRMM Model.

## Examples

```
>>> model = DRMM()
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 5
>>> model.params['mlp_num_fan_out'] = 1
>>> model.params['mlp_activation_func'] = 'tanh'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build**(*self*)

Build model structure.

**forward**(*self*, *inputs*)

Forward.

**class** `matchzoo.models.DRMMTKS`

Bases: `matchzoo.engine.base_model.BaseModel`

DRMMTKS Model.

## Examples

```
>>> model = DRMMTKS()
>>> model.params['top_k'] = 10
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 5
>>> model.params['mlp_num_fan_out'] = 1
>>> model.params['mlp_activation_func'] = 'tanh'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**build**(*self*)

Build model structure.

**forward**(*self*, *inputs*)

Forward.

**class** `matchzoo.models.ESIM`

Bases: `matchzoo.engine.base_model.BaseModel`

ESIM Model.

## Examples

```
>>> model = ESIM()
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build**(*self*)

Instantiating layers.

**forward**(*self*, *inputs*)

Forward.

**class** `matchzoo.models.KNRM`

Bases: `matchzoo.engine.base_model.BaseModel`

KNRM Model.

## Examples

```
>>> model = KNRM()
>>> model.params['kernel_num'] = 11
>>> model.params['sigma'] = 0.1
>>> model.params['exact_sigma'] = 0.001
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```



**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**build**(*self*)

Build model structure.

**forward**(*self*, *inputs*)

Forward.

**class** `matchzoo.models.ConvKNRM`

Bases: `matchzoo.engine.base_model.BaseModel`

ConvKNRM Model.

### Examples

```
>>> model = ConvKNRM()
>>> model.params['filters'] = 128
>>> model.params['conv_activation_func'] = 'tanh'
>>> model.params['max_ngram'] = 3
>>> model.params['use_crossmatch'] = True
>>> model.params['kernel_num'] = 11
>>> model.params['sigma'] = 0.1
>>> model.params['exact_sigma'] = 0.001
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**build**(*self*)

Build model structure.

**forward**(*self*, *inputs*)

Forward.

**class** `matchzoo.models.BiMPM`

Bases: `matchzoo.engine.base_model.BaseModel`

BiMPM Model.

Reference: - <https://github.com/galsang/BIMPM-pytorch/blob/master/model/BIMPM.py>

### Examples

```
>>> model = BiMPM()
>>> model.params['num_perspective'] = 4
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build** (*self*)  
Make function layers.

**forward** (*self*, *inputs*)  
Forward.

**reset\_parameters** (*self*)  
Init Parameters.

**dropout** (*self*, *v*)  
Dropout Layer.

**class** matchzoo.models.**MatchLSTM**  
Bases: *matchzoo.engine.base\_model.BaseModel*

MatchLSTM Model.

<https://github.com/shuohangwang/mprc/blob/master/qa/rankerReader.lua>.

## Examples

```
>>> model = MatchLSTM()
>>> model.params['dropout'] = 0.2
>>> model.params['hidden_size'] = 200
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** **get\_default\_params** (*cls*)

**Returns** model default parameters.

**classmethod** **get\_default\_padding\_callback** (*cls*)

**Returns** Default padding callback.

**build** (*self*)  
Instantiating layers.

**forward** (*self*, *inputs*)  
Forward.

**class** matchzoo.models.**ArcI**  
Bases: *matchzoo.engine.base\_model.BaseModel*

ArcI Model.

## Examples

```
>>> model = ArcI()
>>> model.params['left_filters'] = [32]
>>> model.params['right_filters'] = [32]
>>> model.params['left_kernel_sizes'] = [3]
>>> model.params['right_kernel_sizes'] = [3]
>>> model.params['left_pool_sizes'] = [2]
>>> model.params['right_pool_sizes'] = [4]
>>> model.params['conv_activation_func'] = 'relu'
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 64
>>> model.params['mlp_num_fan_out'] = 32
```

(continues on next page)

(continued from previous page)

```
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.params['dropout_rate'] = 0.5
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build**(*self*)

Build model structure.

ArcI use Siamese arthitecture.

**forward**(*self*, *inputs*)

Forward.

**classmethod** `_make_conv_pool_block(cls, in_channels:int, out_channels:int, kernel_size:int, activation:nn.Module, pool_size:int)`

Make conv pool block.

**class** `matchzoo.models.ArcII`

Bases: `matchzoo.engine.base_model.BaseModel`

ArcII Model.

Examples: `>>> model = ArcII() >>> model.params['embedding_output_dim'] = 300`  
`>>> model.params['kernel_1d_count'] = 32 >>> model.params['kernel_1d_size'] = 3 >>>`  
`model.params['kernel_2d_count'] = [16, 32] >>> model.params['kernel_2d_size'] = [[3, 3], [3, 3]] >>>`  
`model.params['pool_2d_size'] = [[2, 2], [2, 2]] >>> model.guess_and_fill_missing_params(verbose=0) >>>`  
`model.build()`

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**classmethod** `get_default_padding_callback(cls)`

**Returns** Default padding callback.

**build**(*self*)

Build model structure.

ArcII has the desirable property of letting two sentences meet before their own high-level representations mature.

**forward**(*self*, *inputs*)

Forward.

**classmethod** `_make_conv_pool_block(cls, in_channels:int, out_channels:int, kernel_size:tuple, activation:nn.Module, pool_size:tuple)`

Make conv pool block.

**class** `matchzoo.models.Bert`

Bases: `matchzoo.engine.base_model.BaseModel`

Bert Model.

**classmethod** `get_default_params(cls)`

**Returns** model default parameters.

**build** (*self*)

Build model structure.

**forward** (*self*, *inputs*)

Forward.

`matchzoo.models.list_available()` → list

`matchzoo.modules`

## Submodules

`matchzoo.modules.attention`

Attention module.

## Module Contents

**class** `matchzoo.modules.attention.Attention` (*input\_size:int=100, mask:int=0*)

Bases: `torch.nn.Module`

Attention module.

### Parameters

- **input\_size** – Size of input.
- **mask** – An integer to mask the invalid values. Defaults to 0.

## Examples

```
>>> import torch
>>> attention = Attention(input_size=10)
>>> x = torch.randn(4, 5, 10)
>>> x.shape
torch.Size([4, 5, 10])
>>> attention(x).shape
torch.Size([4, 5])
```

**forward** (*self*, *x*)

Perform attention on the input.

**class** `matchzoo.modules.attention.BidirectionalAttention`

Bases: `torch.nn.Module`

Computing the soft attention between two sequence.

**forward** (*self*, *v1*, *v1\_mask*, *v2*, *v2\_mask*)

Forward.

**class** `matchzoo.modules.attention.MatchModule` (*hidden\_size, dropout\_rate=0*)

Bases: `torch.nn.Module`

Computing the match representation for Match LSTM.

### Parameters

- **hidden\_size** – Size of hidden vectors.
- **dropout\_rate** – Dropout rate of the projection layer. Defaults to 0.

## Examples

```
>>> import torch
>>> attention = MatchModule(hidden_size=10)
>>> v1 = torch.randn(4, 5, 10)
>>> v1.shape
torch.Size([4, 5, 10])
>>> v2 = torch.randn(4, 5, 10)
>>> v2_mask = torch.ones(4, 5).to(dtype=torch.uint8)
>>> attention(v1, v2, v2_mask).shape
torch.Size([4, 5, 20])
```

**forward** (*self*, *v1*, *v2*, *v2\_mask*)  
Computing attention vectors and projection vectors.

## matchzoo.modules.bert\_module

Bert module.

## Module Contents

**class** matchzoo.modules.bert\_module.**BertModule** (*mode:str='bert-base-uncased'*)  
Bases: torch.nn.Module

Bert module.

BERT (from Google) released with the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.

**Parameters mode** – String, supported mode can be referred [https://huggingface.co/pytorch-transformers/pretrained\\_models.html](https://huggingface.co/pytorch-transformers/pretrained_models.html).

**forward** (*self*, *x*, *y*)  
Forward.

## matchzoo.modules.dropout

## Module Contents

**class** matchzoo.modules.dropout.**RNNDropout**  
Bases: torch.nn.Dropout

Dropout for RNN.

**forward** (*self*, *sequences\_batch*)  
Masking whole hidden vector for tokens.

## `matchzoo.modules.gaussian_kernel`

Gaussian kernel module.

### Module Contents

```
class matchzoo.modules.gaussian_kernel.GaussianKernel (mu:float=1.0,  
                                                    sigma:float=1.0)
```

Bases: `torch.nn.Module`

Gaussian kernel module.

#### Parameters

- **mu** – Float, mean of the kernel.
- **sigma** – Float, sigma of the kernel.

### Examples

```
>>> import torch
>>> kernel = GaussianKernel()
>>> x = torch.randn(4, 5, 10)
>>> x.shape
torch.Size([4, 5, 10])
>>> kernel(x).shape
torch.Size([4, 5, 10])
```

**forward** (*self, x*)  
Forward.

## `matchzoo.modules.matching`

Matching module.

### Module Contents

```
class matchzoo.modules.matching.Matching (normalize:bool=False, matching_type:str='dot')
```

Bases: `torch.nn.Module`

Module that computes a matching matrix between samples in two tensors.

#### Parameters

- **normalize** – Whether to L2-normalize samples along the dot product axis before taking the dot product. If set to *True*, then the output of the dot product is the cosine proximity between the two samples.
- **matching\_type** – the similarity function for matching

### Examples

```
>>> import torch
>>> matching = Matching(matching_type='dot', normalize=True)
>>> x = torch.randn(2, 3, 2)
>>> y = torch.randn(2, 4, 2)
>>> matching(x, y).shape
torch.Size([2, 3, 4])
```

**classmethod** `_validate_matching_type` (*cls*, *matching\_type*:str='dot')

**forward** (*self*, *x*, *y*)

Perform attention on the input.

`matchzoo.modules.stacked_brnn`

## Module Contents

**class** `matchzoo.modules.stacked_brnn.StackedBRNN` (*input\_size*, *hidden\_size*,  
*num\_layers*, *dropout\_rate*=0,  
*dropout\_output*=False,  
*rnn\_type*=nn.LSTM, *concat\_layers*=False)

Bases: `torch.nn.Module`

Stacked Bi-directional RNNs.

Differs from standard PyTorch library in that it has the option to save and concat the hidden states between layers. (i.e. the output hidden size for each sequence input is `num_layers * hidden_size`).

## Examples

```
>>> import torch
>>> rnn = StackedBRNN(
...     input_size=10,
...     hidden_size=10,
...     num_layers=2,
...     dropout_rate=0.2,
...     dropout_output=True,
...     concat_layers=False
... )
>>> x = torch.randn(2, 5, 10)
>>> x.size()
torch.Size([2, 5, 10])
>>> x_mask = (torch.ones(2, 5) == 1)
>>> rnn(x, x_mask).shape
torch.Size([2, 5, 20])
```

**forward** (*self*, *x*, *x\_mask*)

Encode either padded or non-padded sequences.

**\_forward\_unpadded** (*self*, *x*, *x\_mask*)

Faster encoding that ignores any padding.

## Package Contents

**class** matchzoo.modules.**Attention** (*input\_size:int=100, mask:int=0*)

Bases: torch.nn.Module

Attention module.

### Parameters

- **input\_size** – Size of input.
- **mask** – An integer to mask the invalid values. Defaults to 0.

## Examples

```
>>> import torch
>>> attention = Attention(input_size=10)
>>> x = torch.randn(4, 5, 10)
>>> x.shape
torch.Size([4, 5, 10])
>>> attention(x).shape
torch.Size([4, 5])
```

**forward** (*self, x*)

Perform attention on the input.

**class** matchzoo.modules.**BidirectionalAttention**

Bases: torch.nn.Module

Computing the soft attention between two sequence.

**forward** (*self, v1, v1\_mask, v2, v2\_mask*)

Forward.

**class** matchzoo.modules.**MatchModule** (*hidden\_size, dropout\_rate=0*)

Bases: torch.nn.Module

Computing the match representation for Match LSTM.

### Parameters

- **hidden\_size** – Size of hidden vectors.
- **dropout\_rate** – Dropout rate of the projection layer. Defaults to 0.

## Examples

```
>>> import torch
>>> attention = MatchModule(hidden_size=10)
>>> v1 = torch.randn(4, 5, 10)
>>> v1.shape
torch.Size([4, 5, 10])
>>> v2 = torch.randn(4, 5, 10)
>>> v2_mask = torch.ones(4, 5).to(dtype=torch.uint8)
>>> attention(v1, v2, v2_mask).shape
torch.Size([4, 5, 20])
```

**forward** (*self, v1, v2, v2\_mask*)

Computing attention vectors and projection vectors.



**class** matchzoo.modules.**RNNDropout**

Bases: torch.nn.Dropout

Dropout for RNN.

**forward**(*self*, *sequences\_batch*)

Masking whole hidden vector for tokens.

**class** matchzoo.modules.**StackedBRNN**(*input\_size*, *hidden\_size*, *num\_layers*, *dropout\_rate=0*,  
*dropout\_output=False*, *rnn\_type=nn.LSTM*, *concat\_layers=False*)

Bases: torch.nn.Module

Stacked Bi-directional RNNs.

Differs from standard PyTorch library in that it has the option to save and concat the hidden states between layers. (i.e. the output hidden size for each sequence input is *num\_layers* \* *hidden\_size*).

## Examples

```
>>> import torch
>>> rnn = StackedBRNN(
...     input_size=10,
...     hidden_size=10,
...     num_layers=2,
...     dropout_rate=0.2,
...     dropout_output=True,
...     concat_layers=False
... )
>>> x = torch.randn(2, 5, 10)
>>> x.size()
torch.Size([2, 5, 10])
>>> x_mask = (torch.ones(2, 5) == 1)
>>> rnn(x, x_mask).shape
torch.Size([2, 5, 20])
```

**forward**(*self*, *x*, *x\_mask*)

Encode either padded or non-padded sequences.

**\_forward\_unpadded**(*self*, *x*, *x\_mask*)

Faster encoding that ignores any padding.

**class** matchzoo.modules.**GaussianKernel**(*mu:float=1.0*, *sigma:float=1.0*)

Bases: torch.nn.Module

Gaussian kernel module.

### Parameters

- **mu** – Float, mean of the kernel.
- **sigma** – Float, sigma of the kernel.

## Examples

```
>>> import torch
>>> kernel = GaussianKernel()
>>> x = torch.randn(4, 5, 10)
```

(continues on next page)

(continued from previous page)

```
>>> x.shape
torch.Size([4, 5, 10])
>>> kernel(x).shape
torch.Size([4, 5, 10])
```

**forward** (*self*, *x*)  
Forward.

**class** matchzoo.modules.**Matching** (*normalize:bool=False, matching\_type:str='dot'*)

Bases: torch.nn.Module

Module that computes a matching matrix between samples in two tensors.

#### Parameters

- **normalize** – Whether to L2-normalize samples along the dot product axis before taking the dot product. If set to *True*, then the output of the dot product is the cosine proximity between the two samples.
- **matching\_type** – the similarity function for matching

#### Examples

```
>>> import torch
>>> matching = Matching(matching_type='dot', normalize=True)
>>> x = torch.randn(2, 3, 2)
>>> y = torch.randn(2, 4, 2)
>>> matching(x, y).shape
torch.Size([2, 3, 4])
```

**classmethod** **\_validate\_matching\_type** (*cls, matching\_type:str='dot'*)

**forward** (*self*, *x*, *y*)  
Perform attention on the input.

**class** matchzoo.modules.**BertModule** (*mode:str='bert-base-uncased'*)

Bases: torch.nn.Module

Bert module.

BERT (from Google) released with the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.

**Parameters mode** – String, supported mode can be referred [https://huggingface.co/pytorch-transformers/pretrained\\_models.html](https://huggingface.co/pytorch-transformers/pretrained_models.html).

**forward** (*self*, *x*, *y*)  
Forward.

**matchzoo.preprocessors**

#### Subpackages

**matchzoo.preprocessors.units**

#### Submodules

## matchzoo.preprocessors.units.character\_index

### Module Contents

**class** matchzoo.preprocessors.units.character\_index.CharacterIndex (*char\_index:dict*)  
 Bases: *matchzoo.preprocessors.units.unit.Unit*

CharacterIndexUnit for DIIN model.

The input of :class:'CharacterIndexUnit' should be a list of word character list extracted from a text. The output is the character index representation of this text.

NgramLetterUnit and VocabularyUnit are two essential prerequisite of CharacterIndexUnit.

### Examples

```
>>> input_ = [['#', 'a', '#'], ['#', 'o', 'n', 'e', '#']]
>>> character_index = CharacterIndex(
...     char_index={
...         '<PAD>': 0, '<OOV>': 1, 'a': 2, 'n': 3, 'e':4, '#':5})
>>> index = character_index.transform(input_)
>>> index
[[5, 2, 5], [5, 1, 3, 4, 5]]
```

**transform** (*self, input\_:list*)

Transform list of characters to corresponding indices.

**Parameters** **input** – list of characters generated by :class:'NgramLetterUnit'.

**Returns** character index representation of a text.

## matchzoo.preprocessors.units.digit\_removal

### Module Contents

**class** matchzoo.preprocessors.units.digit\_removal.DigitRemoval  
 Bases: *matchzoo.preprocessors.units.unit.Unit*

Process unit to remove digits.

**transform** (*self, input\_:list*)

Remove digits from list of tokens.

**Parameters** **input** – list of tokens to be filtered.

**Return tokens** tokens of tokens without digits.

## matchzoo.preprocessors.units.frequency\_filter

### Module Contents

**class** matchzoo.preprocessors.units.frequency\_filter.FrequencyFilter (*low:float=0,*  
*high:float=float('inf'),*  
*mode:str='df')*  
 Bases: *matchzoo.preprocessors.units.stateful\_unit.StatefulUnit*

Frequency filter unit.

#### Parameters

- **low** – Lower bound, inclusive.
- **high** – Upper bound, exclusive.
- **mode** – One of *tf* (term frequency), *df* (document frequency), and *idf* (inverse document frequency).

#### Examples::

```
>>> import matchzoo as mz
```

#### To filter based on term frequency (tf):

```
>>> tf_filter = mz.preprocessors.units.FrequencyFilter(  
...     low=2, mode='tf')  
>>> tf_filter.fit(['A', 'B', 'B'], ['C', 'C', 'C'])  
>>> tf_filter.transform(['A', 'B', 'C'])  
['B', 'C']
```

#### To filter based on document frequency (df):

```
>>> tf_filter = mz.preprocessors.units.FrequencyFilter(  
...     low=2, mode='df')  
>>> tf_filter.fit(['A', 'B'], ['B', 'C'])  
>>> tf_filter.transform(['A', 'B', 'C'])  
['B']
```

#### To filter based on inverse document frequency (idf):

```
>>> idf_filter = mz.preprocessors.units.FrequencyFilter(  
...     low=1.2, mode='idf')  
>>> idf_filter.fit(['A', 'B'], ['B', 'C', 'D'])  
>>> idf_filter.transform(['A', 'B', 'C'])  
['A', 'C']
```

**fit** (*self*, *list\_of\_tokens*:*typing.List[typing.List[str]]*)

Fit *list\_of\_tokens* by calculating *mode* states.

**transform** (*self*, *input\_*:*list*)

Transform a list of tokens by filtering out unwanted words.

**classmethod** **\_tf** (*cls*, *list\_of\_tokens*:*list*)

**classmethod** **\_df** (*cls*, *list\_of\_tokens*:*list*)

**classmethod** **\_idf** (*cls*, *list\_of\_tokens*:*list*)

`matchzoo.preprocessors.units.lemmatization`

## Module Contents

**class** `matchzoo.preprocessors.units.lemmatization.Lemmatization`

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for token lemmatization.

**transform** (*self*, *input\_*:list)

Lemmatization a sequence of tokens.

**Parameters** **input** – list of tokens to be lemmatized.

**Return tokens** list of lemmatized tokens.

`matchzoo.preprocessors.units.lowercase`

## Module Contents

**class** `matchzoo.preprocessors.units.lowercase.Lowercase`

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for text lower case.

**transform** (*self*, *input\_*:list)

Convert list of tokens to lower case.

**Parameters** **input** – list of tokens.

**Return tokens** lower-cased list of tokens.

`matchzoo.preprocessors.units.matching_histogram`

## Module Contents

**class** `matchzoo.preprocessors.units.matching_histogram.MatchingHistogram` (*bin\_size*:int=30,  
*embedding\_matrix*:np.ndarray=None,  
*normalize*:bool=True,  
*mode*:str='LCH')

Bases: `matchzoo.preprocessors.units.unit.Unit`

MatchingHistogramUnit Class.

### Parameters

- **bin\_size** – The number of bins of the matching histogram.
- **embedding\_matrix** – The word embedding matrix applied to calculate the matching histogram.
- **normalize** – Boolean, normalize the embedding or not.
- **mode** – The type of the histogram, it should be one of 'CH', 'NG', or 'LCH'.

## Examples

```
>>> embedding_matrix = np.array([[1.0, -1.0], [1.0, 2.0], [1.0, 3.0]])
>>> text_left = [0, 1]
>>> text_right = [1, 2]
>>> histogram = MatchingHistogram(3, embedding_matrix, True, 'CH')
```

(continues on next page)

(continued from previous page)

```
>>> histogram.transform([text_left, text_right])
[[3.0, 1.0, 1.0], [1.0, 2.0, 2.0]]
```

**\_normalize\_embedding**(*self*)  
Normalize the embedding matrix.

**transform**(*self*, *input\_:list*)  
Transform the input text.

`matchzoo.preprocessors.units.ngram_letter`

## Module Contents

**class** `matchzoo.preprocessors.units.ngram_letter.NgramLetter` (*ngram:int=3*, *reduce\_dim:bool=True*)

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for n-letter generation.

Triletter is used in `DSSMModel`. This processor is expected to execute before `Vocab` has been created.

## Examples

```
>>> triletter = NgramLetter()
>>> rv = triletter.transform(['hello', 'word'])
>>> len(rv)
9
>>> rv
['#he', 'hel', 'ell', 'llo', 'lo#', '#wo', 'wor', 'ord', 'rd#']
>>> triletter = NgramLetter(reduce_dim=False)
>>> rv = triletter.transform(['hello', 'word'])
>>> len(rv)
2
>>> rv
[['#he', 'hel', 'ell', 'llo', 'lo#'], ['#wo', 'wor', 'ord', 'rd#']]
```

**transform**(*self*, *input\_:list*)  
Transform token into tri-letter.

For example, *word* should be represented as *#wo*, *wor*, *ord* and *rd#*.

**Parameters** *input* – list of tokens to be transformed.

**Return** *n\_letters* generated *n\_letters*.

`matchzoo.preprocessors.units.punc_removal`

## Module Contents

**class** `matchzoo.preprocessors.units.punc_removal.PuncRemoval`

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for remove punctuations.

**`_MATCH_PUNC`****`transform`** (*self*, *input\_*:list)

Remove punctuations from list of tokens.

**Parameters** *input* – list of tokens.**Return** *rv* tokens without punctuation.**`matchzoo.preprocessors.units.stateful_unit`****Module Contents****class** `matchzoo.preprocessors.units.stateful_unit.StatefulUnit`Bases: `matchzoo.preprocessors.units.unit.Unit`

Unit with inner state.

Usually need to be fit before transforming. All information gathered in the fit phrase will be stored into its *context*.**state**Get current context. Same as *unit.context*.Deprecated since v2.2.0, and will be removed in the future. Used *unit.context* instead.**context**Get current context. Same as *unit.state*.**fit** (*self*, *input\_*:typing.Any)

Abstract base method, need to be implemented in subclass.

**`matchzoo.preprocessors.units.stemming`****Module Contents****class** `matchzoo.preprocessors.units.stemming.Stemming` (*stemmer*='porter')Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for token stemming.

**Parameters** *stemmer* – stemmer to use, *porter* or *lancaster*.**transform** (*self*, *input\_*:list)

Reducing inflected words to their word stem, base or root form.

**Parameters** *input* – list of string to be stemmed.**`matchzoo.preprocessors.units.stop_removal`****Module Contents****class** `matchzoo.preprocessors.units.stop_removal.StopRemoval` (*lang*:str='english')Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit to remove stop words.

### Example

```
>>> unit = StopRemoval()
>>> unit.transform(['a', 'the', 'test'])
['test']
>>> type(unit.stopwords)
<class 'list'>
```

#### **stopwords**

Get stopwords based on language.

**Params** **lang** language code.

**Returns** list of stop words.

**transform** (*self*, *input\_:list*)

Remove stopwords from list of tokenized tokens.

#### **Parameters**

- **input** – list of tokenized tokens.
- **lang** – language code for stopwords.

**Return tokens** list of tokenized tokens without stopwords.

`matchzoo.preprocessors.units.tokenize`

### Module Contents

**class** `matchzoo.preprocessors.units.tokenize.Tokenize`

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for text tokenization.

**transform** (*self*, *input\_:str*)

Process input data from raw terms to list of tokens.

**Parameters** **input** – raw textual input.

**Return tokens** tokenized tokens as a list.

`matchzoo.preprocessors.units.truncated_length`

### Module Contents

**class** `matchzoo.preprocessors.units.truncated_length.TruncatedLength` (*text\_length:int*,  
*truncate\_mode:str='pre'*)

Bases: `matchzoo.preprocessors.units.unit.Unit`

TruncatedLengthUnit Class.

Process unit to truncate the text that exceeds the set length.



## Examples

```
>>> from matchzoo.preprocessors.units import TruncatedLength
>>> truncatedlen = TruncatedLength(3)
>>> truncatedlen.transform(list(range(1, 6))) == [3, 4, 5]
True
>>> truncatedlen.transform(list(range(2))) == [0, 1]
True
```

**transform** (*self*, *input\_:list*)

Truncate the text that exceeds the specified maximum length.

**Parameters** *input* – list of tokenized tokens.

**Return tokens** list of tokenized tokens in fixed length if its origin length larger than *text\_length*.

`matchzoo.preprocessors.units.unit`

## Module Contents

**class** `matchzoo.preprocessors.units.unit.Unit`

Process unit do not persive state (i.e. do not need fit).

**transform** (*self*, *input\_:typing.Any*)

Abstract base method, need to be implemented in subclass.

`matchzoo.preprocessors.units.vocabulary`

## Module Contents

**class** `matchzoo.preprocessors.units.vocabulary.Vocabulary` (*pad\_value:str='<PAD>'*,  
*oov\_value:str='<OOV>'*)

Bases: `matchzoo.preprocessors.units.stateful_unit.StatefulUnit`

Vocabulary class.

**Parameters**

- **pad\_value** – The string value for the padding position.
- **oov\_value** – The string value for the out-of-vocabulary terms.

## Examples

```
>>> vocab = Vocabulary(pad_value='[PAD]', oov_value='[OOV]')
>>> vocab.fit(['A', 'B', 'C', 'D', 'E'])
>>> term_index = vocab.state['term_index']
>>> term_index # doctest: +SKIP
{'[PAD]': 0, '[OOV]': 1, 'D': 2, 'A': 3, 'B': 4, 'C': 5, 'E': 6}
>>> index_term = vocab.state['index_term']
>>> index_term # doctest: +SKIP
{0: '[PAD]', 1: '[OOV]', 2: 'D', 3: 'A', 4: 'B', 5: 'C', 6: 'E'}
```

```

>>> term_index['out-of-vocabulary-term']
1
>>> index_term[0]
'[PAD]'
>>> index_term[42]
Traceback (most recent call last):
...
KeyError: 42
>>> a_index = term_index['A']
>>> c_index = term_index['C']
>>> vocab.transform(['C', 'A', 'C']) == [c_index, a_index, c_index]
True
>>> vocab.transform(['C', 'A', '[OOV]']) == [c_index, a_index, 1]
True
>>> indices = vocab.transform(list('ABCDDZZZ'))
>>> ' '.join(vocab.state['index_term'][i] for i in indices)
'A B C D D [OOV] [OOV] [OOV]'

```

#### **class TermIndex**

Bases: dict

Map term to index.

**\_\_missing\_\_**(self, key)

Map out-of-vocabulary terms to index 1.

**fit**(self, tokens:list)

Build a *TermIndex* and a *IndexTerm*.

**transform**(self, input\_:list)

Transform a list of tokens to corresponding indices.

#### **matchzoo.preprocessors.units.word\_exact\_match**

### Module Contents

**class** matchzoo.preprocessors.units.word\_exact\_match.**WordExactMatch**(match:str,  
to\_match:str)

Bases: *matchzoo.preprocessors.units.unit.Unit*

WordExactUnit Class.

Process unit to get a binary match list of two word index lists. The word index list is the word representation of a text.

### Examples

```

>>> import pandas
>>> input_ = pandas.DataFrame({
...   'text_left':[[1, 2, 3],[4, 5, 7, 9]],
...   'text_right':[[5, 3, 2, 7],[2, 3, 5]]
... })
>>> left_word_exact_match = WordExactMatch(
...     match='text_left', to_match='text_right'
... )
>>> left_out = input_.apply(left_word_exact_match.transform, axis=1)

```

(continues on next page)

(continued from previous page)

```

>>> left_out[0]
[0, 1, 1]
>>> left_out[1]
[0, 1, 0, 0]
>>> right_word_exact_match = WordExactMatch(
...     match='text_right', to_match='text_left'
... )
>>> right_out = input_.apply(right_word_exact_match.transform, axis=1)
>>> right_out[0]
[0, 1, 1, 0]
>>> right_out[1]
[0, 0, 1]

```

**transform** (*self*, *input\_*)

Transform two word index lists into a binary match list.

**Parameters** *input* – a dataframe include ‘match’ column and ‘to\_match’ column.

**Returns** a binary match result list of two word index lists.

## matchzoo.preprocessors.units.word\_hashing

### Module Contents

**class** matchzoo.preprocessors.units.word\_hashing.**WordHashing** (*term\_index:dict*)

Bases: *matchzoo.preprocessors.units.unit.Unit*

Word-hashing layer for DSSM-based models.

The input of WordHashingUnit should be a list of word sub-letter list extracted from one document. The output of is the word-hashing representation of this document.

NgramLetterUnit and VocabularyUnit are two essential prerequisite of WordHashingUnit.

### Examples

```

>>> letters = [['#te', 'tes', 'est', 'st#'], ['oov']]
>>> word_hashing = WordHashing(
...     term_index={
...         '_PAD': 0, 'OOV': 1, 'st#': 2, '#te': 3, 'est': 4, 'tes': 5
...     })
>>> hashing = word_hashing.transform(letters)
>>> hashing[0]
[0.0, 0.0, 1.0, 1.0, 1.0, 1.0]
>>> hashing[1]
[0.0, 1.0, 0.0, 0.0, 0.0, 0.0]

```

**transform** (*self*, *input\_:list*)

Transform list of letters into word hashing layer.

**Parameters** *input* – list of *tri\_letters* generated by NgramLetterUnit.

**Returns** Word hashing representation of *tri-letters*.

## Package Contents

**class** matchzoo.preprocessors.units.**Unit**

Process unit do not persive state (i.e. do not need fit).

**transform** (*self*, *input\_:typing.Any*)

Abstract base method, need to be implemented in subclass.

**class** matchzoo.preprocessors.units.**DigitRemoval**

Bases: *matchzoo.preprocessors.units.unit.Unit*

Process unit to remove digits.

**transform** (*self*, *input\_:list*)

Remove digits from list of tokens.

**Parameters** **input** – list of tokens to be filtered.

**Return tokens** tokens of tokens without digits.

**class** matchzoo.preprocessors.units.**FrequencyFilter** (*low:float=0*,  
*high:float=float('inf')*,  
*mode:str='df'*)

Bases: *matchzoo.preprocessors.units.stateful\_unit.StatefulUnit*

Frequency filter unit.

### Parameters

- **low** – Lower bound, inclusive.
- **high** – Upper bound, exclusive.
- **mode** – One of *tf* (term frequency), *df* (document frequency), and *idf* (inverse document frequency).

### Examples::

```
>>> import matchzoo as mz
```

### To filter based on term frequency (tf):

```
>>> tf_filter = mz.preprocessors.units.FrequencyFilter(  
...     low=2, mode='tf')  
>>> tf_filter.fit(['A', 'B', 'B'], ['C', 'C', 'C'])  
>>> tf_filter.transform(['A', 'B', 'C'])  
['B', 'C']
```

### To filter based on document frequency (df):

```
>>> tf_filter = mz.preprocessors.units.FrequencyFilter(  
...     low=2, mode='df')  
>>> tf_filter.fit(['A', 'B'], ['B', 'C'])  
>>> tf_filter.transform(['A', 'B', 'C'])  
['B']
```

### To filter based on inverse document frequency (idf):

```
>>> idf_filter = mz.preprocessors.units.FrequencyFilter(  
...     low=1.2, mode='idf')  
>>> idf_filter.fit(['A', 'B'], ['B', 'C', 'D'])
```

(continues on next page)

(continued from previous page)

```
>>> idf_filter.transform(['A', 'B', 'C'])
['A', 'C']
```

**fit** (*self*, *list\_of\_tokens*:*typing.List[typing.List[str]]*)

Fit *list\_of\_tokens* by calculating *mode* states.

**transform** (*self*, *input\_*:*list*)

Transform a list of tokens by filtering out unwanted words.

**classmethod** **\_tf** (*cls*, *list\_of\_tokens*:*list*)

**classmethod** **\_df** (*cls*, *list\_of\_tokens*:*list*)

**classmethod** **\_idf** (*cls*, *list\_of\_tokens*:*list*)

**class** matchzoo.preprocessors.units.**Lemma****matization**

Bases: *matchzoo.preprocessors.units.unit.Unit*

Process unit for token lemmatization.

**transform** (*self*, *input\_*:*list*)

Lemmatization a sequence of tokens.

**Parameters** **input** – list of tokens to be lemmatized.

**Return tokens** list of lemmatized tokens.

**class** matchzoo.preprocessors.units.**Lowercase**

Bases: *matchzoo.preprocessors.units.unit.Unit*

Process unit for text lower case.

**transform** (*self*, *input\_*:*list*)

Convert list of tokens to lower case.

**Parameters** **input** – list of tokens.

**Return tokens** lower-cased list of tokens.

**class** matchzoo.preprocessors.units.**MatchingHistogram** (*bin\_size*:*int*=30, *embedding\_matrix*:*None*, *normalize*:*True*, *mode*:*str*='LCH')

Bases: *matchzoo.preprocessors.units.unit.Unit*

MatchingHistogramUnit Class.

**Parameters**

- **bin\_size** – The number of bins of the matching histogram.
- **embedding\_matrix** – The word embedding matrix applied to calculate the matching histogram.
- **normalize** – Boolean, normalize the embedding or not.
- **mode** – The type of the histogram, it should be one of 'CH', 'NG', or 'LCH'.

## Examples

```
>>> embedding_matrix = np.array([[1.0, -1.0], [1.0, 2.0], [1.0, 3.0]])
>>> text_left = [0, 1]
>>> text_right = [1, 2]
>>> histogram = MatchingHistogram(3, embedding_matrix, True, 'CH')
>>> histogram.transform([text_left, text_right])
[[3.0, 1.0, 1.0], [1.0, 2.0, 2.0]]
```

**\_normalize\_embedding**(*self*)

Normalize the embedding matrix.

**transform**(*self*, *input\_:list*)

Transform the input text.

**class** matchzoo.preprocessors.units.**NgramLetter**(*ngram:int=3, reduce\_dim:bool=True*)

Bases: *matchzoo.preprocessors.units.unit.Unit*

Process unit for n-letter generation.

Triletter is used in DSSMModel. This processor is expected to execute before *Vocab* has been created.

## Examples

```
>>> triletter = NgramLetter()
>>> rv = triletter.transform(['hello', 'word'])
>>> len(rv)
9
>>> rv
['#he', 'hel', 'ell', 'llo', 'lo#', '#wo', 'wor', 'ord', 'rd#']
>>> triletter = NgramLetter(reduce_dim=False)
>>> rv = triletter.transform(['hello', 'word'])
>>> len(rv)
2
>>> rv
[['#he', 'hel', 'ell', 'llo', 'lo#'], ['#wo', 'wor', 'ord', 'rd#']]
```

**transform**(*self*, *input\_:list*)

Transform token into tri-letter.

For example, *word* should be represented as *#wo*, *wor*, *ord* and *rd#*.

**Parameters** *input* – list of tokens to be transformed.

**Return** *n\_letters* generated *n\_letters*.

**class** matchzoo.preprocessors.units.**PuncRemoval**

Bases: *matchzoo.preprocessors.units.unit.Unit*

Process unit for remove punctuations.

**\_MATCH\_PUNC**

**transform**(*self*, *input\_:list*)

Remove punctuations from list of tokens.

**Parameters** *input* – list of tokens.

**Return** *rv* tokens without punctuation.

**class** matchzoo.preprocessors.units.**StatefulUnit**

Bases: *matchzoo.preprocessors.units.unit.Unit*

Unit with inner state.

Usually need to be fit before transforming. All information gathered in the fit phrase will be stored into its *context*.

#### **state**

Get current context. Same as *unit.context*.

Deprecated since v2.2.0, and will be removed in the future. Used *unit.context* instead.

#### **context**

Get current context. Same as *unit.state*.

**fit** (*self*, *input\_*:*typing.Any*)

Abstract base method, need to be implemented in subclass.

**class** `matchzoo.preprocessors.units.Stemming` (*stemmer*='porter')

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for token stemming.

**Parameters** **stemmer** – stemmer to use, *porter* or *lancaster*.

**transform** (*self*, *input\_*:*list*)

Reducing inflected words to their word stem, base or root form.

**Parameters** **input** – list of string to be stemmed.

**class** `matchzoo.preprocessors.units.StopRemoval` (*lang*:*str*='english')

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit to remove stop words.

### Example

```
>>> unit = StopRemoval()
>>> unit.transform(['a', 'the', 'test'])
['test']
>>> type(unit.stopwords)
<class 'list'>
```

#### **stopwords**

Get stopwords based on language.

**Params** **lang** language code.

**Returns** list of stop words.

**transform** (*self*, *input\_*:*list*)

Remove stopwords from list of tokenized tokens.

#### **Parameters**

- **input** – list of tokenized tokens.
- **lang** – language code for stopwords.

**Return tokens** list of tokenized tokens without stopwords.

**class** `matchzoo.preprocessors.units.Tokenize`

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for text tokenization.

**transform** (*self*, *input\_:str*)

Process input data from raw terms to list of tokens.

**Parameters** *input* – raw textual input.

**Return** *tokens* tokenized tokens as a list.

**class** matchzoo.preprocessors.units.**Vocabulary** (*pad\_value:str='<PAD>'*,  
*oov\_value:str='<OOV>'*)  
 Bases: *matchzoo.preprocessors.units.stateful\_unit.StatefulUnit*

Vocabulary class.

**Parameters**

- **pad\_value** – The string value for the padding position.
- **oov\_value** – The string value for the out-of-vocabulary terms.

## Examples

```
>>> vocab = Vocabulary(pad_value='[PAD]', oov_value='[OOV]')
>>> vocab.fit(['A', 'B', 'C', 'D', 'E'])
>>> term_index = vocab.state['term_index']
>>> term_index # doctest: +SKIP
{'[PAD]': 0, '[OOV]': 1, 'D': 2, 'A': 3, 'B': 4, 'C': 5, 'E': 6}
>>> index_term = vocab.state['index_term']
>>> index_term # doctest: +SKIP
{0: '[PAD]', 1: '[OOV]', 2: 'D', 3: 'A', 4: 'B', 5: 'C', 6: 'E'}
```

```
>>> term_index['out-of-vocabulary-term']
1
>>> index_term[0]
'[PAD]'
>>> index_term[42]
Traceback (most recent call last):
...
KeyError: 42
>>> a_index = term_index['A']
>>> c_index = term_index['C']
>>> vocab.transform(['C', 'A', 'C']) == [c_index, a_index, c_index]
True
>>> vocab.transform(['C', 'A', '[OOV]']) == [c_index, a_index, 1]
True
>>> indices = vocab.transform(list('ABCDDZZZ'))
>>> ' '.join(vocab.state['index_term'][i] for i in indices)
'A B C D D [OOV] [OOV] [OOV]'
```

**class** **TermIndex**

Bases: dict

Map term to index.

**\_\_missing\_\_** (*self*, *key*)

Map out-of-vocabulary terms to index 1.

**fit** (*self*, *tokens:list*)

Build a *TermIndex* and a *IndexTerm*.

**transform** (*self*, *input\_:list*)

Transform a list of tokens to corresponding indices.



**class** matchzoo.preprocessors.units.**WordHashing** (*term\_index:dict*)

Bases: *matchzoo.preprocessors.units.unit.Unit*

Word-hashing layer for DSSM-based models.

The input of WordHashingUnit should be a list of word sub-letter list extracted from one document. The output of is the word-hashing representation of this document.

NgramLetterUnit and VocabularyUnit are two essential prerequisite of WordHashingUnit.

### Examples

```
>>> letters = [['#te', 'tes', 'est', 'st#'], ['oov']]
>>> word_hashing = WordHashing(
...     term_index={
...         '_PAD': 0, 'OOV': 1, 'st#': 2, '#te': 3, 'est': 4, 'tes': 5
...     })
>>> hashing = word_hashing.transform(letters)
>>> hashing[0]
[0.0, 0.0, 1.0, 1.0, 1.0, 1.0]
>>> hashing[1]
[0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
```

**transform** (*self, input\_:list*)

Transform list of letters into word hashing layer.

**Parameters** **input** – list of *tri\_letters* generated by NgramLetterUnit.

**Returns** Word hashing representation of *tri-letters*.

**class** matchzoo.preprocessors.units.**CharacterIndex** (*char\_index:dict*)

Bases: *matchzoo.preprocessors.units.unit.Unit*

CharacterIndexUnit for DIIN model.

The input of :class:'CharacterIndexUnit' should be a list of word character list extracted from a text. The output is the character index representation of this text.

NgramLetterUnit and VocabularyUnit are two essential prerequisite of CharacterIndexUnit.

### Examples

```
>>> input_ = [['#', 'a', '#'], ['#', 'o', 'n', 'e', '#']]
>>> character_index = CharacterIndex(
...     char_index={
...         '<PAD>': 0, '<OOV>': 1, 'a': 2, 'n': 3, 'e': 4, '#': 5})
>>> index = character_index.transform(input_)
>>> index
[[5, 2, 5], [5, 1, 3, 4, 5]]
```

**transform** (*self, input\_:list*)

Transform list of characters to corresponding indices.

**Parameters** **input** – list of characters generated by :class:'NgramLetterUnit'.

**Returns** character index representation of a text.

**class** matchzoo.preprocessors.units.**WordExactMatch** (*match:str, to\_match:str*)

Bases: *matchzoo.preprocessors.units.unit.Unit*

WordExactUnit Class.

Process unit to get a binary match list of two word index lists. The word index list is the word representation of a text.

### Examples

```
>>> import pandas
>>> input_ = pandas.DataFrame({
...     'text_left':[[1, 2, 3],[4, 5, 7, 9]],
...     'text_right':[[5, 3, 2, 7],[2, 3, 5]]
... })
>>> left_word_exact_match = WordExactMatch(
...     match='text_left', to_match='text_right'
... )
>>> left_out = input_.apply(left_word_exact_match.transform, axis=1)
>>> left_out[0]
[0, 1, 1]
>>> left_out[1]
[0, 1, 0, 0]
>>> right_word_exact_match = WordExactMatch(
...     match='text_right', to_match='text_left'
... )
>>> right_out = input_.apply(right_word_exact_match.transform, axis=1)
>>> right_out[0]
[0, 1, 1, 0]
>>> right_out[1]
[0, 0, 1]
```

**transform**(*self*, *input\_*)

Transform two word index lists into a binary match list.

**Parameters** *input* – a dataframe include ‘match’ column and ‘to\_match’ column.

**Returns** a binary match result list of two word index lists.

**class** matchzoo.preprocessors.units.**TruncatedLength**(*text\_length:int*, *truncate\_mode:str='pre'*) *truncate*

Bases: *matchzoo.preprocessors.units.unit.Unit*

TruncatedLengthUnit Class.

Process unit to truncate the text that exceeds the set length.

### Examples

```
>>> from matchzoo.preprocessors.units import TruncatedLength
>>> truncatedlen = TruncatedLength(3)
>>> truncatedlen.transform(list(range(1, 6))) == [3, 4, 5]
True
>>> truncatedlen.transform(list(range(2))) == [0, 1]
True
```

**transform**(*self*, *input\_:list*)

Truncate the text that exceeds the specified maximum length.

**Parameters** *input* – list of tokenized tokens.

**Return tokens** list of tokenized tokens in fixed length if its origin length larger than `text_length`.

`matchzoo.preprocessors.units.list_available()` → list

## Submodules

`matchzoo.preprocessors.basic_preprocessor`

Basic Preprocessor.

## Module Contents

```
class matchzoo.preprocessors.basic_preprocessor.BasicPreprocessor (truncated_mode:str='pre',
truncated_length_left:int=30,
truncated_length_right:int=30,
filter_mode:str='df',
filter_low_freq:float=1,
filter_high_freq:float=float('inf'),
remove_stop_words:bool=False)
```

Bases: `matchzoo.engine.base_preprocessor.BasePreprocessor`

Basic preprocessor helper.

### Parameters

- **truncated\_mode** – String, mode used by `TruncatedLength`. Can be ‘pre’ or ‘post’.
- **truncated\_length\_left** – Integer, maximize length of left in the data\_pack.
- **truncated\_length\_right** – Integer, maximize length of right in the data\_pack.
- **filter\_mode** – String, mode used by `FrequencyFilterUnit`. Can be ‘df’, ‘cf’, and ‘idf’.
- **filter\_low\_freq** – Float, lower bound value used by `FrequencyFilterUnit`.
- **filter\_high\_freq** – Float, upper bound value used by `FrequencyFilterUnit`.
- **remove\_stop\_words** – Bool, use `StopRemovalUnit` unit or not.

## Example

```
>>> import matchzoo as mz
>>> train_data = mz.datasets.toy.load_data('train')
>>> test_data = mz.datasets.toy.load_data('test')
>>> preprocessor = mz.preprocessors.BasicPreprocessor(
...     truncated_length_left=10,
...     truncated_length_right=20,
...     filter_mode='df',
```

(continues on next page)

(continued from previous page)

```

...     filter_low_freq=2,
...     filter_high_freq=1000,
...     remove_stop_words=True
... )
>>> preprocessor = preprocessor.fit(train_data, verbose=0)
>>> preprocessor.context['vocab_size']
226
>>> processed_train_data = preprocessor.transform(train_data,
...                                              verbose=0)
>>> type(processed_train_data)
<class 'matchzoo.data_pack.data_pack.DataPack'>
>>> test_data_transformed = preprocessor.transform(test_data,
...                                              verbose=0)
>>> type(test_data_transformed)
<class 'matchzoo.data_pack.data_pack.DataPack'>

```

**fit** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Fit pre-processing context for transformation.

#### Parameters

- **data\_pack** – data\_pack to be preprocessed.
- **verbose** – Verbosity.

**Returns** class:*BasicPreprocessor* instance.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data, create truncated length representation.

#### Parameters

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as *DataPack* object.

## matchzoo.preprocessors.bert\_preprocessor

Bert Preprocessor.

## Module Contents

**class** matchzoo.preprocessors.bert\_preprocessor.**BertPreprocessor** (*mode:str='bert-base-uncased'*)

Bases: *matchzoo.engine.base\_preprocessor.BasePreprocessor*

Baisc preprocessor helper.

**Parameters** **mode** – String, supported mode can be referred [https://huggingface.co/pytorch-transformers/pretrained\\_models.html](https://huggingface.co/pytorch-transformers/pretrained_models.html).

**fit** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Tokenizer is all BertPreprocessor's need.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data.

**Parameters**

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as `DataPack` object.

`matchzoo.preprocessors.build_unit_from_data_pack`

Build unit from data pack.

**Module Contents**

`matchzoo.preprocessors.build_unit_from_data_pack.build_unit_from_data_pack` (*unit:StatefulUnit, data\_pack:mz.DataPack, mode:str='both', flatten:bool=True, verbose:int=1*)  
 →  
 StatefulUnit

Build a `StatefulUnit` from a `DataPack` object.

**Parameters**

- **unit** – `StatefulUnit` object to be built.
- **data\_pack** – The input `DataPack` object.
- **mode** – One of 'left', 'right', and 'both', to determine the source data for building the `VocabularyUnit`.
- **flatten** – Flatten the datapack or not. *True* to organize the `DataPack` text as a list, and *False* to organize `DataPack` text as a list of list.
- **verbose** – Verbosity.

**Returns** A built `StatefulUnit` object.

`matchzoo.preprocessors.build_vocab_unit`

**Module Contents**

`matchzoo.preprocessors.build_vocab_unit.build_vocab_unit` (*data\_pack:DataPack, mode:str='both', verbose:int=1*) → `Vocabulary`

Build a `preprocessor.units.Vocabulary` given *data\_pack*.

The *data\_pack* should be preprocessed beforehand, and each item in *text\_left* and *text\_right* columns of the *data\_pack* should be a list of tokens.

### Parameters

- **data\_pack** – The `DataPack` to build vocabulary upon.
- **mode** – One of ‘left’, ‘right’, and ‘both’, to determine the source

data for building the `VocabularyUnit`. :param verbose: Verbosity. :return: A built vocabulary unit.

### `matchzoo.preprocessors.cdssm_preprocessor`

CDSSM Preprocessor.

### Module Contents

```
class matchzoo.preprocessors.cdssm_preprocessor.CDSSMPreprocessor(truncated_mode:str='pre',  
                                                                trun-  
                                                                cated_length_left:int=10,  
                                                                trun-  
                                                                cated_length_right:int=40,  
                                                                with_word_hashing:bool=True)
```

Bases: `matchzoo.engine.base_preprocessor.BasePreprocessor`

CDSSM Model preprocessor.

**with\_word\_hashing**  
*with\_word\_hashing* getter.

**fit** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Fit pre-processing context for transformation.

### Parameters

- **verbose** – Verbosity.
- **data\_pack** – Data\_pack to be preprocessed.

**Returns** class:`CDSSMPreprocessor` instance.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data, create *letter-ngram* representation.

### Parameters

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as `DataPack` object.

**classmethod \_default\_units** (*cls*)  
Prepare needed process units.

### `matchzoo.preprocessors.chain_transform`

Wrapper function organizes a number of transform functions.

## Module Contents

`matchzoo.preprocessors.chain_transform.chain_transform` (*units:typing.List[Unit]*) → *typing.Callable*

Compose unit transformations into a single function.

**Parameters** *units* – List of `matchzoo.StatelessUnit`.

## `matchzoo.preprocessors.diin_preprocessor`

DIIN Preprocessor.

## Module Contents

**class** `matchzoo.preprocessors.diin_preprocessor.DIINPreprocessor` (*truncated\_mode:str='pre', truncated\_length\_left:int=30, truncated\_length\_right:int=50*)

Bases: `matchzoo.engine.base_preprocessor.BasePreprocessor`

DIIN Model preprocessor.

**fit** (*self, data\_pack:DataPack, verbose:int=1*)  
Fit pre-processing context for transformation.

### Parameters

- **data\_pack** – data\_pack to be preprocessed.
- **verbose** – Verbosity.

**Returns** class:'DIINPreprocessor' instance.

**transform** (*self, data\_pack:DataPack, verbose:int=1*)  
Apply transformation on data.

### Parameters

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as :class:'DataPack' object.

## `matchzoo.preprocessors.dssm_preprocessor`

DSSM Preprocessor.

## Module Contents

**class** `matchzoo.preprocessors.dssm_preprocessor.DSSMPreprocessor` (*with\_word\_hashing:bool=True*)  
Bases: `matchzoo.engine.base_preprocessor.BasePreprocessor`

DSSM Model preprocessor.

**with\_word\_hashing**

*with\_word\_hashing* getter.

**fit** (*self*, *data\_pack*:*DataPack*, *verbose*:*int*=1)  
Fit pre-processing context for transformation.

**Parameters**

- **verbose** – Verbosity.
- **data\_pack** – *data\_pack* to be preprocessed.

**Returns** class:*DSSMPreprocessor* instance.

**transform** (*self*, *data\_pack*:*DataPack*, *verbose*:*int*=1)  
Apply transformation on data, create *tri-letter* representation.

**Parameters**

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as *DataPack* object.

**classmethod** **\_default\_units** (*cls*)  
Prepare needed process units.

`matchzoo.preprocessors.naive_preprocessor`

Naive Preprocessor.

## Module Contents

**class** `matchzoo.preprocessors.naive_preprocessor.NaivePreprocessor`

Bases: `matchzoo.engine.base_preprocessor.BasePreprocessor`

Naive preprocessor.

## Example

```
>>> import matchzoo as mz
>>> train_data = mz.datasets.toy.load_data()
>>> test_data = mz.datasets.toy.load_data(stage='test')
>>> preprocessor = mz.preprocessors.NaivePreprocessor()
>>> train_data_processed = preprocessor.fit_transform(train_data,
...                                                  verbose=0)
>>> type(train_data_processed)
<class 'matchzoo.data_pack.data_pack.DataPack'>
>>> test_data_transformed = preprocessor.transform(test_data,
...                                              verbose=0)
>>> type(test_data_transformed)
<class 'matchzoo.data_pack.data_pack.DataPack'>
```

**fit** (*self*, *data\_pack*:*DataPack*, *verbose*:*int*=1)  
Fit pre-processing context for transformation.

**Parameters**



- **data\_pack** – data\_pack to be preprocessed.
- **verbose** – Verbosity.

**Returns** class:*NaivePreprocessor* instance.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data, create truncated length representation.

**Parameters**

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as *DataPack* object.

## Package Contents

**class** matchzoo.preprocessors.**DSSMPreprocessor** (*with\_word\_hashing:bool=True*)

Bases: *matchzoo.engine.base\_preprocessor.BasePreprocessor*

DSSM Model preprocessor.

**with\_word\_hashing**  
*with\_word\_hashing* getter.

**fit** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Fit pre-processing context for transformation.

**Parameters**

- **verbose** – Verbosity.
- **data\_pack** – data\_pack to be preprocessed.

**Returns** class:*DSSMPreprocessor* instance.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data, create *tri-letter* representation.

**Parameters**

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as *DataPack* object.

**classmethod** **\_default\_units** (*cls*)  
Prepare needed process units.

**class** matchzoo.preprocessors.**NaivePreprocessor**

Bases: *matchzoo.engine.base\_preprocessor.BasePreprocessor*

Naive preprocessor.

## Example

```
>>> import matchzoo as mz
>>> train_data = mz.datasets.toy.load_data()
>>> test_data = mz.datasets.toy.load_data(stage='test')
>>> preprocessor = mz.preprocessors.NaivePreprocessor()
>>> train_data_processed = preprocessor.fit_transform(train_data,
...                                                  verbose=0)
>>> type(train_data_processed)
<class 'matchzoo.data_pack.data_pack.DataPack'>
>>> test_data_transformed = preprocessor.transform(test_data,
...                                              verbose=0)
>>> type(test_data_transformed)
<class 'matchzoo.data_pack.data_pack.DataPack'>
```

**fit** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Fit pre-processing context for transformation.

**Parameters**

- **data\_pack** – data\_pack to be preprocessed.
- **verbose** – Verbosity.

**Returns** class:*NaivePreprocessor* instance.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data, create truncated length representation.

**Parameters**

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as *DataPack* object.

```
class matchzoo.preprocessors.BasicPreprocessor (truncated_mode:str='pre',          trun-
                                                cated_length_left:int=30,          trun-
                                                cated_length_right:int=30,
                                                filter_mode:str='df',          fil-
                                                ter_low_freq:float=1,          fil-
                                                ter_high_freq:float=float('inf'), re-
                                                move_stop_words:bool=False)
```

Bases: *matchzoo.engine.base\_preprocessor.BasePreprocessor*

Basic preprocessor helper.

**Parameters**

- **truncated\_mode** – String, mode used by *TruncatedLength*. Can be ‘pre’ or ‘post’.
- **truncated\_length\_left** – Integer, maximize length of left in the data\_pack.
- **truncated\_length\_right** – Integer, maximize length of right in the data\_pack.
- **filter\_mode** – String, mode used by *FrequencyFilterUnit*. Can be ‘df’, ‘cf’, and ‘idf’.
- **filter\_low\_freq** – Float, lower bound value used by *FrequencyFilterUnit*.
- **filter\_high\_freq** – Float, upper bound value used by *FrequencyFilterUnit*.
- **remove\_stop\_words** – Bool, use *StopRemovalUnit* unit or not.

## Example

```
>>> import matchzoo as mz
>>> train_data = mz.datasets.toy.load_data('train')
>>> test_data = mz.datasets.toy.load_data('test')
>>> preprocessor = mz.preprocessors.BasicPreprocessor(
...     truncated_length_left=10,
...     truncated_length_right=20,
...     filter_mode='df',
...     filter_low_freq=2,
...     filter_high_freq=1000,
...     remove_stop_words=True
... )
>>> preprocessor = preprocessor.fit(train_data, verbose=0)
>>> preprocessor.context['vocab_size']
226
>>> processed_train_data = preprocessor.transform(train_data,
...                                              verbose=0)
>>> type(processed_train_data)
<class 'matchzoo.data_pack.data_pack.DataPack'>
>>> test_data_transformed = preprocessor.transform(test_data,
...                                              verbose=0)
>>> type(test_data_transformed)
<class 'matchzoo.data_pack.data_pack.DataPack'>
```

**fit** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Fit pre-processing context for transformation.

### Parameters

- **data\_pack** – data\_pack to be preprocessed.
- **verbose** – Verbosity.

**Returns** class:*BasicPreprocessor* instance.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data, create truncated length representation.

### Parameters

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as DataPack object.

```
class matchzoo.preprocessors.CDSSMPreprocessor (truncated_mode:str='pre',          trun-
                                                cated_length_left:int=10,      trun-
                                                cated_length_right:int=40,
                                                with_word_hashing:bool=True)
```

Bases: *matchzoo.engine.base\_preprocessor.BasePreprocessor*

CDSSM Model preprocessor.

**with\_word\_hashing**  
*with\_word\_hashing* getter.

**fit** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Fit pre-processing context for transformation.

### Parameters

- **verbose** – Verbosity.
- **data\_pack** – Data\_pack to be preprocessed.

**Returns** class: *CDSSMPreprocessor* instance.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data, create *letter-ngram* representation.

**Parameters**

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as *DataPack* object.

**classmethod** **\_default\_units** (*cls*)  
Prepare needed process units.

**class** *matchzoo.preprocessors.DIINPreprocessor* (*truncated\_mode:str='pre', truncated\_length\_left:int=30, truncated\_length\_right:int=50*)

Bases: *matchzoo.engine.base\_preprocessor.BasePreprocessor*

DIIN Model preprocessor.

**fit** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Fit pre-processing context for transformation.

**Parameters**

- **data\_pack** – data\_pack to be preprocessed.
- **verbose** – Verbosity.

**Returns** class: 'DIINPreprocessor' instance.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data.

**Parameters**

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as :class: 'DataPack' object.

**class** *matchzoo.preprocessors.BertPreprocessor* (*mode:str='bert-base-uncased'*)

Bases: *matchzoo.engine.base\_preprocessor.BasePreprocessor*

Baisc preprocessor helper.

**Parameters** **mode** – String, supported mode can be referred [https://huggingface.co/pytorch-transformers/pretrained\\_models.html](https://huggingface.co/pytorch-transformers/pretrained_models.html).

**fit** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Tokenizer is all BertPreprocessor's need.

**transform** (*self*, *data\_pack:DataPack*, *verbose:int=1*)  
Apply transformation on data.

**Parameters**

- **data\_pack** – Inputs to be preprocessed.
- **verbose** – Verbosity.

**Returns** Transformed data as DataPack object.

`matchzoo.preprocessors.list_available()` → list

`matchzoo.tasks`

## Submodules

`matchzoo.tasks.classification`

Classification task.

## Module Contents

**class** `matchzoo.tasks.classification.Classification` (*num\_classes: int=2, \*\*kwargs*)

Bases: `matchzoo.engine.base_task.BaseTask`

Classification task.

## Examples

```
>>> classification_task = Classification(num_classes=2)
>>> classification_task.metrics = ['acc']
>>> classification_task.num_classes
2
>>> classification_task.output_shape
(2,)
>>> classification_task.output_dtype
<class 'int'>
>>> print(classification_task)
Classification Task with 2 classes
```

**TYPE = classification**

**num\_classes**

number of classes to classify.

**Type** return

**output\_shape**

output shape of a single sample of the task.

**Type** return

**output\_dtype**

target data type, expect *int* as output.

**Type** return

**classmethod** `list_available_losses` (*cls*)

**Returns** a list of available losses.

**classmethod** `list_available_metrics` (*cls*)

**Returns** a list of available metrics.

**\_\_str\_\_** (*self*)

**Returns** Task name as string.

`matchzoo.tasks.ranking`

Ranking task.

## Module Contents

**class** `matchzoo.tasks.ranking.Ranking`  
Bases: `matchzoo.engine.base_task.BaseTask`  
Ranking Task.

## Examples

```
>>> ranking_task = Ranking()
>>> ranking_task.metrics = ['map', 'ndcg']
>>> ranking_task.output_shape
(1,)
>>> ranking_task.output_dtype
<class 'float'>
>>> print(ranking_task)
Ranking Task
```

**TYPE = ranking**

**output\_shape**  
output shape of a single sample of the task.

**Type** return

**output\_dtype**  
target data type, expect *float* as output.

**Type** return

**classmethod list\_available\_losses** (*cls*)

**Returns** a list of available losses.

**classmethod list\_available\_metrics** (*cls*)

**Returns** a list of available metrics.

**\_\_str\_\_** (*self*)

**Returns** Task name as string.

## Package Contents

**class** `matchzoo.tasks.Classification` (*num\_classes:int=2, \*\*kwargs*)  
Bases: `matchzoo.engine.base_task.BaseTask`  
Classification task.

## Examples

```
>>> classification_task = Classification(num_classes=2)
>>> classification_task.metrics = ['acc']
>>> classification_task.num_classes
2
>>> classification_task.output_shape
(2,)
>>> classification_task.output_dtype
<class 'int'>
>>> print(classification_task)
Classification Task with 2 classes
```

**TYPE = classification**

**num\_classes**

number of classes to classify.

**Type** return

**output\_shape**

output shape of a single sample of the task.

**Type** return

**output\_dtype**

target data type, expect *int* as output.

**Type** return

**classmethod list\_available\_losses** (*cls*)

**Returns** a list of available losses.

**classmethod list\_available\_metrics** (*cls*)

**Returns** a list of available metrics.

**\_\_str\_\_** (*self*)

**Returns** Task name as string.

**class** matchzoo.tasks.**Ranking**

Bases: *matchzoo.engine.base\_task.BaseTask*

Ranking Task.

## Examples

```
>>> ranking_task = Ranking()
>>> ranking_task.metrics = ['map', 'ndcg']
>>> ranking_task.output_shape
(1,)
>>> ranking_task.output_dtype
<class 'float'>
>>> print(ranking_task)
Ranking Task
```

**TYPE = ranking**

**output\_shape**

output shape of a single sample of the task.

**Type** return

**output\_dtype**

target data type, expect *float* as output.

**Type** return

**classmethod** `list_available_losses` (*cls*)

**Returns** a list of available losses.

**classmethod** `list_available_metrics` (*cls*)

**Returns** a list of available metrics.

**\_\_str\_\_** (*self*)

**Returns** Task name as string.

`matchzoo.trainers`

## Submodules

`matchzoo.trainers.trainer`

Base Trainer.

## Module Contents

**class** `matchzoo.trainers.trainer.Trainer` (*model:BaseModel*, *optimizer:optim.Optimizer*, *trainloader:DataLoader*, *validloader:DataLoader*, *device:typing.Optional[torch.device]=None*, *start\_epoch:int=1*, *epochs:int=10*, *validate\_interval:typing.Optional[int]=None*, *scheduler:typing.Any=None*, *clip\_norm:typing.Union[float, int]=None*, *patience:typing.Optional[int]=None*, *key:typing.Any=None*, *data\_parallel:bool=True*, *checkpoint:typing.Union[str, Path]=None*, *save\_dir:typing.Union[str, Path]=None*, *save\_all:bool=False*, *verbose:int=1*, *\*\*kwargs*)

MatchZoo tranier.

### Parameters

- **model** – A `BaseModel` instance.
- **optimizer** – A `optim.Optimizer` instance.
- **trainloader** – A `:class'DataLoader'` instance. The dataloader is used for training the model.
- **validloader** – A `:class'DataLoader'` instance. The dataloader is used for validating the model.
- **device** – The desired device of returned tensor. Default: if `None`, uses the current device for the default tensor type (see `torch.set_default_tensor_type()`). device will be the CPU for CPU tensor types and the current CUDA device for CUDA tensor types.



- **start\_epoch** – Int. Number of starting epoch.
- **epochs** – The maximum number of epochs for training. Defaults to 10.
- **validate\_interval** – Int. Interval of validation.
- **scheduler** – LR scheduler used to adjust the learning rate based on the number of epochs.
- **clip\_norm** – Max norm of the gradients to be clipped.
- **patience** – Number of events to wait if no improvement and then stop the training.
- **key** – Key of metric to be compared.
- **data\_parallel** – Bool. Whether support data parallel.
- **checkpoint** – A checkpoint from which to continue training. If None, training starts from scratch. Defaults to None. Should be a file-like object (has to implement read, readline, tell, and seek), or a string containing a file name.
- **save\_dir** – Directory to save trainer.
- **save\_all** – Bool. If True, save *Trainer* instance; If False, only save model. Defaults to False.
- **verbose** – 0, 1, or 2. Verbosity mode. 0 = silent, 1 = verbose, 2 = one log line per epoch.

**\_load\_data\_loader** (*self*, *trainloader:DataLoader*, *validloader:DataLoader*, *validate\_interval:typing.Optional[int]=None*)  
Load trainloader and determine validate interval.

#### Parameters

- **trainloader** – A :class:'DataLoader' instance. The dataloader is used to train the model.
- **validloader** – A :class:'DataLoader' instance. The dataloader is used to validate the model.
- **validate\_interval** – int. Interval of validation.

**\_load\_model** (*self*, *model:BaseModel*, *device:typing.Optional[torch.device]*, *data\_parallel:bool=True*)  
Load model.

#### Parameters

- **model** – BaseModel instance.
- **device** – the desired device of returned tensor. Default: if None, uses the current device for the default tensor type (see torch.set\_default\_tensor\_type()). device will be the CPU for CPU tensor types and the current CUDA device for CUDA tensor types.
- **data\_parallel** – bool. Whether support data parallel.

**\_load\_path** (*self*, *checkpoint:typing.Union[str, Path]*, *save\_dir:typing.Union[str, Path]*)  
Load save\_dir and Restore from checkpoint.

#### Parameters

- **checkpoint** – A checkpoint from which to continue training. If None, training starts from scratch. Defaults to None. Should be a file-like object (has to implement read, readline, tell, and seek), or a string containing a file name.
- **save\_dir** – Directory to save trainer.

**`_backward`** (*self*, *loss*)

Computes the gradient of current *loss* graph leaves.

**Parameters** **`loss`** – Tensor. Loss of model.

**`_run_scheduler`** (*self*)

Run scheduler.

**`run`** (*self*)

Train model.

**The processes:** Run each epoch -> Run scheduler -> Should stop early?

**`_run_epoch`** (*self*)

Run each epoch.

**The training steps:**

- Get batch and feed them into model
- Get outputs. Caculate all losses and sum them up
- Loss backwards and optimizer steps
- Evaluation
- Update and output result

**`evaluate`** (*self*, *dataloader: DataLoader*)

Evaluate the model.

**Parameters** **`dataloader`** – A DataLoader object to iterate over the data.

**classmethod** **`_eval_metric_on_data_frame`** (*cls*, *metric: BaseMetric*, *id\_left: typing.Any*,  
*y\_true: typing.Union[list, np.array]*,  
*y\_pred: typing.Union[list, np.array]*)

Eval metric on data frame.

This function is used to eval metrics for *Ranking* task.

**Parameters**

- **`metric`** – Metric for *Ranking* task.
- **`id_left`** – id of input left. Samples with same `id_left` should be grouped for evaluation.
- **`y_true`** – Labels of dataset.
- **`y_pred`** – Outputs of model.

**Returns** Evaluation result.

**`predict`** (*self*, *dataloader: DataLoader*)

Generate output predictions for the input samples.

**Parameters** **`dataloader`** – input DataLoader

**Returns** predictions

**`_save`** (*self*)

Save.

**`save_model`** (*self*)

Save the model.

**`save`** (*self*)

Save the trainer.

*Trainer* parameters like epoch, best\_so\_far, model, optimizer and early\_stopping will be saved to specific file path.

**Parameters path** – Path to save trainer.

**restore\_model** (*self*, *checkpoint:typing.Union[str, Path]*)  
Restore model.

**Parameters checkpoint** – A checkpoint from which to continue training.

**restore** (*self*, *checkpoint:typing.Union[str, Path]=None*)  
Restore trainer.

**Parameters checkpoint** – A checkpoint from which to continue training.

## Package Contents

```
class matchzoo.trainers.Trainer(model:BaseModel, optimizer:optim.Optimizer, train-  
                                loader:DataLoader, validloader:DataLoader, de-  
                                vice:typing.Optional[torch.device]=None, start_epoch:int=1,  
                                epochs:int=10, validate_interval:typing.Optional[int]=None,  
                                scheduler:typing.Any=None, clip_norm:typing.Union[float,  
                                int]=None, patience:typing.Optional[int]=None,  
                                key:typing.Any=None, data_parallel:bool=True,  
                                checkpoint:typing.Union[str, Path]=None,  
                                save_dir:typing.Union[str, Path]=None, save_all:bool=False,  
                                verbose:int=1, **kwargs)
```

MatchZoo trainer.

### Parameters

- **model** – A BaseModel instance.
- **optimizer** – A optim.Optimizer instance.
- **trainloader** – A :class'Dataloader' instance. The dataloader is used for training the model.
- **validloader** – A :class'Dataloader' instance. The dataloader is used for validating the model.
- **device** – The desired device of returned tensor. Default: if None, uses the current device for the default tensor type (see torch.set\_default\_tensor\_type()). device will be the CPU for CPU tensor types and the current CUDA device for CUDA tensor types.
- **start\_epoch** – Int. Number of starting epoch.
- **epochs** – The maximum number of epochs for training. Defaults to 10.
- **validate\_interval** – Int. Interval of validation.
- **scheduler** – LR scheduler used to adjust the learning rate based on the number of epochs.
- **clip\_norm** – Max norm of the gradients to be clipped.
- **patience** – Number of events to wait if no improvement and then stop the training.
- **key** – Key of metric to be compared.
- **data\_parallel** – Bool. Whether support data parallel.

- **checkpoint** – A checkpoint from which to continue training. If None, training starts from scratch. Defaults to None. Should be a file-like object (has to implement read, readline, tell, and seek), or a string containing a file name.
- **save\_dir** – Directory to save trainer.
- **save\_all** – Bool. If True, save *Trainer* instance; If False, only save model. Defaults to False.
- **verbose** – 0, 1, or 2. Verbosity mode. 0 = silent, 1 = verbose, 2 = one log line per epoch.

**\_load\_data\_loader** (*self*, *trainloader:DataLoader*, *validloader:DataLoader*, *validate\_interval:typing.Optional[int]=None*)  
Load trainloader and determine validate interval.

#### Parameters

- **trainloader** – A :class‘DataLoader‘ instance. The dataloader is used to train the model.
- **validloader** – A :class‘DataLoader‘ instance. The dataloader is used to validate the model.
- **validate\_interval** – int. Interval of validation.

**\_load\_model** (*self*, *model:BaseModel*, *device:typing.Optional[torch.device]*, *data\_parallel:bool=True*)  
Load model.

#### Parameters

- **model** – BaseModel instance.
- **device** – the desired device of returned tensor. Default: if None, uses the current device for the default tensor type (see torch.set\_default\_tensor\_type()). device will be the CPU for CPU tensor types and the current CUDA device for CUDA tensor types.
- **data\_parallel** – bool. Whether support data parallel.

**\_load\_path** (*self*, *checkpoint:typing.Union[str, Path]*, *save\_dir:typing.Union[str, Path]*)  
Load save\_dir and Restore from checkpoint.

#### Parameters

- **checkpoint** – A checkpoint from which to continue training. If None, training starts from scratch. Defaults to None. Should be a file-like object (has to implement read, readline, tell, and seek), or a string containing a file name.
- **save\_dir** – Directory to save trainer.

**\_backward** (*self*, *loss*)  
Computes the gradient of current *loss* graph leaves.

**Parameters** **loss** – Tensor. Loss of model.

**\_run\_scheduler** (*self*)  
Run scheduler.

**run** (*self*)  
Train model.

**The processes:** Run each epoch -> Run scheduler -> Should stop early?

**\_run\_epoch** (*self*)  
Run each epoch.

**The training steps:**

- Get batch and feed them into model
- Get outputs. Caculate all losses and sum them up
- Loss backwards and optimizer steps
- Evaluation
- Update and output result

**evaluate** (*self*, *dataloader: DataLoader*)

Evaluate the model.

**Parameters** **dataloader** – A DataLoader object to iterate over the data.

**classmethod** **\_eval\_metric\_on\_data\_frame** (*cls*, *metric: BaseMetric*, *id\_left: typing.Any*,  
*y\_true: typing.Union[list, np.array]*,  
*y\_pred: typing.Union[list, np.array]*)

Eval metric on data frame.

This function is used to eval metrics for *Ranking* task.

**Parameters**

- **metric** – Metric for *Ranking* task.
- **id\_left** – id of input left. Samples with same id\_left should be grouped for evaluation.
- **y\_true** – Labels of dataset.
- **y\_pred** – Outputs of model.

**Returns** Evaluation result.

**predict** (*self*, *dataloader: DataLoader*)

Generate output predictions for the input samples.

**Parameters** **dataloader** – input DataLoader

**Returns** predictions

**\_save** (*self*)

Save.

**save\_model** (*self*)

Save the model.

**save** (*self*)

Save the trainer.

*Trainer* parameters like epoch, best\_so\_far, model, optimizer and early\_stopping will be savad to specific file path.

**Parameters** **path** – Path to save trainer.

**restore\_model** (*self*, *checkpoint: typing.Union[str, Path]*)

Restore model.

**Parameters** **checkpoint** – A checkpoint from which to continue training.

**restore** (*self*, *checkpoint: typing.Union[str, Path]=None*)

Restore trainer.

**Parameters** **checkpoint** – A checkpoint from which to continue training.

`matchzoo.utils`

## Submodules

`matchzoo.utils.average_meter`

Average meter.

## Module Contents

**class** `matchzoo.utils.average_meter.AverageMeter`

Bases: `object`

Computes and stores the average and current value.

## Examples

```
>>> am = AverageMeter()
>>> am.update(1)
>>> am.avg
1.0
>>> am.update(val=2.5, n=2)
>>> am.avg
2.0
```

**avg**

Get avg.

**reset** (*self*)

Reset AverageMeter.

**update** (*self*, *val*, *n=1*)

Update value.

`matchzoo.utils.early_stopping`

Early stopping.

## Module Contents

**class** `matchzoo.utils.early_stopping.EarlyStopping` (*patience:typing.Optional[int]=None*,  
*should\_decrease:bool=None*,  
*key:typing.Any=None*)

EarlyStopping stops training if no improvement after a given patience.

### Parameters

- **patience** – Number fo events to wait if no improvement and then stop the training.
- **should\_decrease** – The way to judge the best so far.
- **key** – Key of metric to be compared.

**best\_so\_far**

Returns best so far.

**is\_best\_so\_far**

Returns true if it is the best so far.

**should\_stop\_early**

Returns true if improvement has stopped for long enough.

**state\_dict** (*self*)

A *Trainer* can use this to serialize the state.

**load\_state\_dict** (*self*, *state\_dict*:*typing.Dict[str, typing.Any]*)

Hydrate a early stopping from a serialized state.

**update** (*self*, *result*:*list*)

Call function.

**matchzoo.utils.get\_file**

Download file.

## Module Contents

**class** matchzoo.utils.get\_file.**Progbar** (*target*, *width*=30, *verbose*=1, *interval*=0.05)

Bases: object

Displays a progress bar.

### Parameters

- **target** – Total number of steps expected, None if unknown.
- **width** – Progress bar width on screen.
- **verbose** – Verbosity mode, 0 (silent), 1 (verbose), 2 (semi-verbose)
- **stateful\_metrics** – Iterable of string names of metrics that should *not* be averaged over time. Metrics in this list will be displayed as-is. All others will be averaged by the progbar before display.
- **interval** – Minimum visual progress update interval (in seconds).

**update** (*self*, *current*)

Updates the progress bar.

matchzoo.utils.get\_file.**\_extract\_archive** (*file\_path*, *path*='.', *archive\_format*='auto')

Extracts an archive if it matches tar, tar.gz, tar.bz, or zip formats.

### Parameters

- **file\_path** – path to the archive file
- **path** – path to extract the archive file
- **archive\_format** – Archive format to try for extracting the file. Options are 'auto', 'tar', 'zip', and None. 'tar' includes tar, tar.gz, and tar.bz files. The default 'auto' is ['tar', 'zip']. None or an empty list will return no matches found.

**Returns** True if a match was found and an archive extraction was completed, False otherwise.

```
matchzoo.utils.get_file.get_file(fname:str=None,    origin:str=None,    untar:bool=False,
                                extract:bool=False,    md5_hash:typing.Any=None,
                                file_hash:typing.Any=None,    hash_algorithm:str='auto',
                                archive_format:str='auto',    cache_subdir:typing.Union[Path,
                                str]='data',    cache_dir:typing.Union[Path,
                                str]=matchzoo.USER_DATA_DIR,    verbose:int=1)    →
                                str
```

Downloads a file from a URL if it not already in the cache.

By default the file at the url *origin* is downloaded to the cache\_dir `~/.matchzoo/datasets`, placed in the cache\_subdir *data*, and given the filename *fname*. The final location of a file *example.txt* would therefore be `~/.matchzoo/datasets/data/example.txt`.

Files in tar, tar.gz, tar.bz, and zip formats can also be extracted. Passing a hash will verify the file after download. The command line programs *shasum* and *sha256sum* can compute the hash.

#### Parameters

- **fname** – Name of the file. If an absolute path `/path/to/file.txt` is specified the file will be saved at that location.
- **origin** – Original URL of the file.
- **untar** – Deprecated in favor of ‘extract’. Boolean, whether the file should be decompressed.
- **md5\_hash** – Deprecated in favor of ‘file\_hash’. md5 hash of the file for verification.
- **file\_hash** – The expected hash string of the file after download. The sha256 and md5 hash algorithms are both supported.
- **cache\_subdir** – Subdirectory under the cache dir where the file is saved. If an absolute path `/path/to/folder` is specified the file will be saved at that location.
- **hash\_algorithm** – Select the hash algorithm to verify the file. options are ‘md5’, ‘sha256’, and ‘auto’. The default ‘auto’ detects the hash algorithm in use.
- **archive\_format** – Archive format to try for extracting the file. Options are ‘auto’, ‘tar’, ‘zip’, and None. ‘tar’ includes tar, tar.gz, and tar.bz files. The default ‘auto’ is [‘tar’, ‘zip’]. None or an empty list will return no matches found.
- **cache\_dir** – Location to store cached files, when None it defaults to the `[matchzoo.USER_DATA_DIR](~/.matchzoo/datasets)`.
- **verbose** – Verbosity mode, 0 (silent), 1 (verbose), 2 (semi-verbose)

**Papram extract** True tries extracting the file as an Archive, like tar or zip.

**Returns** Path to the downloaded file.

```
matchzoo.utils.get_file.validate_file(fpath,    file_hash,    algorithm='auto',
                                     chunk_size=65535)
```

Validates a file against a sha256 or md5 hash.

#### Parameters

- **fpath** – path to the file being validated
- **file\_hash** – The expected hash string of the file. The sha256 and md5 hash algorithms are both supported.
- **algorithm** – Hash algorithm, one of ‘auto’, ‘sha256’, or ‘md5’. The default ‘auto’ detects the hash algorithm in use.
- **chunk\_size** – Bytes to read at a time, important for large files.



**Returns** Whether the file is valid.

`matchzoo.utils.get_file._hash_file(fpath, algorithm='sha256', chunk_size=65535)`  
 Calculates a file sha256 or md5 hash.

**Parameters**

- **fpath** – path to the file being validated
- **algorithm** – hash algorithm, one of 'auto', 'sha256', or 'md5'. The default 'auto' detects the hash algorithm in use.
- **chunk\_size** – Bytes to read at a time, important for large files.

**Returns** The file hash.

`matchzoo.utils.list_recursive_subclasses`

## Module Contents

`matchzoo.utils.list_recursive_subclasses.list_recursive_concrete_subclasses(base)`  
 List all concrete subclasses of *base* recursively.

`matchzoo.utils.list_recursive_subclasses._filter_concrete(classes)`

`matchzoo.utils.list_recursive_subclasses._bfs(base)`

`matchzoo.utils.one_hot`

One hot vectors.

## Module Contents

`matchzoo.utils.one_hot.one_hot(indices:int, num_classes:int) → np.ndarray`

**Returns** A one-hot encoded vector.

`matchzoo.utils.parse`

## Module Contents

`matchzoo.utils.parse.activation`

`matchzoo.utils.parse.loss`

`matchzoo.utils.parse.optimizer`

`matchzoo.utils.parse._parse(identifier:typing.Union[str, typing.Type[nn.Module], nn.Module], dictionary:nn.ModuleDict, target:str) → nn.Module`

Parse loss and activation.

**Parameters**

- **identifier** – activation identifier, one of - String: name of a activation - Torch Module subclass - Torch Module instance (it will be returned unchanged).
- **dictionary** – nn.ModuleDict instance. Map string identifier to nn.Module instance.

**Returns** A `nn.Module` instance

`matchzoo.utils.parse.parse_activation` (*identifier:typing.Union[str, typing.Type[nn.Module], nn.Module]*)  $\rightarrow$  `nn.Module`

Retrieves a torch Module instance.

**Parameters** **identifier** – activation identifier, one of - String: name of a activation - Torch Module subclass - Torch Module instance (it will be returned unchanged).

**Returns** A `nn.Module` instance

**Examples::**

```
>>> from torch import nn
>>> from matchzoo.utils import parse_activation
```

**Use `str` as activation:**

```
>>> activation = parse_activation('relu')
>>> type(activation)
<class 'torch.nn.modules.activation.ReLU'>
```

**Use `torch.nn.Module` subclasses as activation:**

```
>>> type(parse_activation(nn.ReLU))
<class 'torch.nn.modules.activation.ReLU'>
```

**Use `torch.nn.Module` instances as activation:**

```
>>> type(parse_activation(nn.ReLU()))
<class 'torch.nn.modules.activation.ReLU'>
```

`matchzoo.utils.parse.parse_loss` (*identifier:typing.Union[str, typing.Type[nn.Module], nn.Module], task:typing.Optional[str]=None*)  $\rightarrow$  `nn.Module`

Retrieves a torch Module instance.

**Parameters**

- **identifier** – loss identifier, one of - String: name of a loss - Torch Module subclass - Torch Module instance (it will be returned unchanged).
- **task** – Task type for determining specific loss.

**Returns** A `nn.Module` instance

**Examples::**

```
>>> from torch import nn
>>> from matchzoo.utils import parse_loss
```

**Use `str` as loss:**

```
>>> loss = parse_loss('mse')
>>> type(loss)
<class 'torch.nn.modules.loss.MSELoss'>
```

**Use `torch.nn.Module` subclasses as loss:**

```
>>> type(parse_loss(nn.MSELoss))
<class 'torch.nn.modules.loss.MSELoss'>
```

Use `torch.nn.Module` instances as loss:

```
>>> type(parse_loss(nn.MSELoss()))
<class 'torch.nn.modules.loss.MSELoss'>
```

`matchzoo.utils.parse._parse_metric` (*metric*:`typing.Union[str, typing.Type[BaseMetric], BaseMetric]`, *Metrix*:`typing.Type[BaseMetric]`) → `BaseMetric`

Parse metric.

#### Parameters

- **metric** – Input metric in any form.
- **Metrix** – Base Metric class. Either `matchzoo.engine.base_metric.RankingMetric` or `matchzoo.engine.base_metric.ClassificationMetric`.

**Returns** A `BaseMetric` instance

`matchzoo.utils.parse.parse_metric` (*metric*:`typing.Union[str, typing.Type[BaseMetric], BaseMetric]`, *task*:`str`) → `BaseMetric`

Parse input metric in any form into a `BaseMetric` instance.

#### Parameters

- **metric** – Input metric in any form.
- **task** – Task type for determining specific metric.

**Returns** A `BaseMetric` instance

Examples::

```
>>> from matchzoo import metrics
>>> from matchzoo.utils import parse_metric
```

Use *str* as MatchZoo metrics:

```
>>> mz_metric = parse_metric('map', 'ranking')
>>> type(mz_metric)
<class 'matchzoo.metrics.mean_average_precision.MeanAveragePrecision'>
```

Use `matchzoo.engine.BaseMetric` subclasses as MatchZoo metrics:

```
>>> type(parse_metric(metrics.AveragePrecision, 'ranking'))
<class 'matchzoo.metrics.average_precision.AveragePrecision'>
```

Use `matchzoo.engine.BaseMetric` instances as MatchZoo metrics:

```
>>> type(parse_metric(metrics.AveragePrecision(), 'ranking'))
<class 'matchzoo.metrics.average_precision.AveragePrecision'>
```

`matchzoo.utils.parse.parse_optimizer` (*identifier*:`typing.Union[str, typing.Type[optim.Optimizer]]`) → `optim.Optimizer`

Parse input metric in any form into a `Optimizer` class.

**Parameters** **optimizer** – Input optimizer in any form.

**Returns** A `Optimizer` class

Examples::

```
>>> from torch import optim
>>> from matchzoo.utils import parse_optimizer
```

Use *str* as optimizer:

```
>>> parse_optimizer('adam')
<class 'torch.optim.adam.Adam'>
```

Use `torch.optim.Optimizer` subclasses as optimizer:

```
>>> parse_optimizer(optim.Adam)
<class 'torch.optim.adam.Adam'>
```

## `matchzoo.utils.tensor_type`

Define Keras tensor type.

### Module Contents

`matchzoo.utils.tensor_type.TensorType`

## `matchzoo.utils.timer`

Timer.

### Module Contents

**class** `matchzoo.utils.timer.Timer`

Bases: `object`

Computes elapsed time.

**time**

Return time.

**reset** (*self*)

Reset timer.

**resume** (*self*)

Resume.

**stop** (*self*)

Stop.

### Package Contents

`matchzoo.utils.one_hot` (*indices:int, num\_classes:int*) → `np.ndarray`

**Returns** A one-hot encoded vector.

`matchzoo.utils.TensorType`

`matchzoo.utils.list_recursive_concrete_subclasses` (*base*)

List all concrete subclasses of *base* recursively.

`matchzoo.utils.parse_loss` (*identifier:typing.Union[str, typing.Type[nn.Module], nn.Module], task:typing.Optional[str]=None*) → *nn.Module*

Retrieves a torch Module instance.

#### Parameters

- **identifier** – loss identifier, one of - String: name of a loss - Torch Module subclass - Torch Module instance (it will be returned unchanged).
- **task** – Task type for determining specific loss.

**Returns** A *nn.Module* instance

#### Examples::

```
>>> from torch import nn
>>> from matchzoo.utils import parse_loss
```

#### Use *str* as loss:

```
>>> loss = parse_loss('mse')
>>> type(loss)
<class 'torch.nn.modules.loss.MSELoss'>
```

#### Use `torch.nn.Module` subclasses as loss:

```
>>> type(parse_loss(nn.MSELoss))
<class 'torch.nn.modules.loss.MSELoss'>
```

#### Use `torch.nn.Module` instances as loss:

```
>>> type(parse_loss(nn.MSELoss()))
<class 'torch.nn.modules.loss.MSELoss'>
```

`matchzoo.utils.parse_activation` (*identifier:typing.Union[str, typing.Type[nn.Module], nn.Module]*) → *nn.Module*

Retrieves a torch Module instance.

**Parameters** **identifier** – activation identifier, one of - String: name of a activation - Torch Module subclass - Torch Module instance (it will be returned unchanged).

**Returns** A *nn.Module* instance

#### Examples::

```
>>> from torch import nn
>>> from matchzoo.utils import parse_activation
```

#### Use *str* as activation:

```
>>> activation = parse_activation('relu')
>>> type(activation)
<class 'torch.nn.modules.activation.ReLU'>
```

#### Use `torch.nn.Module` subclasses as activation:

```
>>> type(parse_activation(nn.ReLU()))
<class 'torch.nn.modules.activation.ReLU'>
```

Use `torch.nn.Module` instances as activation:

```
>>> type(parse_activation(nn.ReLU()))
<class 'torch.nn.modules.activation.ReLU'>
```

`matchzoo.utils.parse_metric` (*metric:typing.Union[str, typing.Type[BaseMetric], BaseMetric], task:str*) → `BaseMetric`  
Parse input metric in any form into a `BaseMetric` instance.

**Parameters**

- **metric** – Input metric in any form.
- **task** – Task type for determining specific metric.

**Returns** A `BaseMetric` instance

**Examples::**

```
>>> from matchzoo import metrics
>>> from matchzoo.utils import parse_metric
```

Use `str` as MatchZoo metrics:

```
>>> mz_metric = parse_metric('map', 'ranking')
>>> type(mz_metric)
<class 'matchzoo.metrics.mean_average_precision.MeanAveragePrecision'>
```

Use `matchzoo.engine.BaseMetric` subclasses as MatchZoo metrics:

```
>>> type(parse_metric(metrics.AveragePrecision, 'ranking'))
<class 'matchzoo.metrics.average_precision.AveragePrecision'>
```

Use `matchzoo.engine.BaseMetric` instances as MatchZoo metrics:

```
>>> type(parse_metric(metrics.AveragePrecision(), 'ranking'))
<class 'matchzoo.metrics.average_precision.AveragePrecision'>
```

`matchzoo.utils.parse_optimizer` (*identifier:typing.Union[str, typing.Type[optim.Optimizer]]*) → `optim.Optimizer`  
Parse input metric in any form into a `Optimizer` class.

**Parameters** **optimizer** – Input optimizer in any form.

**Returns** A `Optimizer` class

**Examples::**

```
>>> from torch import optim
>>> from matchzoo.utils import parse_optimizer
```

Use `str` as optimizer:

```
>>> parse_optimizer('adam')
<class 'torch.optim.adam.Adam'>
```

Use `torch.optim.Optimizer` subclasses as optimizer:

```
>>> parse_optimizer(optimizer.Adam)
<class 'torch.optim.adam.Adam'>
```

**class** `matchzoo.utils.AverageMeter`

Bases: `object`

Computes and stores the average and current value.

### Examples

```
>>> am = AverageMeter()
>>> am.update(1)
>>> am.avg
1.0
>>> am.update(val=2.5, n=2)
>>> am.avg
2.0
```

**avg**

Get avg.

**reset** (*self*)

Reset AverageMeter.

**update** (*self*, *val*, *n=1*)

Update value.

**class** `matchzoo.utils.Timer`

Bases: `object`

Computes elapsed time.

**time**

Return time.

**reset** (*self*)

Reset timer.

**resume** (*self*)

Resume.

**stop** (*self*)

Stop.

**class** `matchzoo.utils.EarlyStopping` (*patience:typing.Optional[int]=None*,  
*should\_decrease:bool=None*, *key:typing.Any=None*)

EarlyStopping stops training if no improvement after a given patience.

**Parameters**

- **patience** – Number of events to wait if no improvement and then stop the training.
- **should\_decrease** – The way to judge the best so far.
- **key** – Key of metric to be compared.

**best\_so\_far**

Returns best so far.

**is\_best\_so\_far**

Returns true if it is the best so far.

**should\_stop\_early**

Returns true if improvement has stopped for long enough.

**state\_dict** (*self*)

A *Trainer* can use this to serialize the state.

**load\_state\_dict** (*self*, *state\_dict*:*typing.Dict[str, typing.Any]*)

Hydrate a early stopping from a serialized state.

**update** (*self*, *result*:*list*)

Call function.

```
matchzoo.utils.get_file (fname:str=None,          origin:str=None,          untar:bool=False,
                        extract:bool=False,        md5_hash:typing.Any=None,
                        file_hash:typing.Any=None,  hash_algorithm:str='auto',
                        archive_format:str='auto',  cache_subdir:typing.Union[Path, str]='data',
                        cache_dir:typing.Union[Path, str]=matchzoo.USER_DATA_DIR,  ver-
                       bose:int=1) → str
```

Downloads a file from a URL if it not already in the cache.

By default the file at the url *origin* is downloaded to the *cache\_dir* `~/.matchzoo/datasets`, placed in the *cache\_subdir* *data*, and given the filename *fname*. The final location of a file *example.txt* would therefore be `~/.matchzoo/datasets/data/example.txt`.

Files in tar, tar.gz, tar.bz, and zip formats can also be extracted. Passing a hash will verify the file after download. The command line programs *shasum* and *sha256sum* can compute the hash.

**Parameters**

- **fname** – Name of the file. If an absolute path `/path/to/file.txt` is specified the file will be saved at that location.
- **origin** – Original URL of the file.
- **untar** – Deprecated in favor of ‘extract’. Boolean, whether the file should be decompressed.
- **md5\_hash** – Deprecated in favor of ‘file\_hash’. md5 hash of the file for verification.
- **file\_hash** – The expected hash string of the file after download. The sha256 and md5 hash algorithms are both supported.
- **cache\_subdir** – Subdirectory under the cache dir where the file is saved. If an absolute path `/path/to/folder` is specified the file will be saved at that location.
- **hash\_algorithm** – Select the hash algorithm to verify the file. options are ‘md5’, ‘sha256’, and ‘auto’. The default ‘auto’ detects the hash algorithm in use.
- **archive\_format** – Archive format to try for extracting the file. Options are ‘auto’, ‘tar’, ‘zip’, and None. ‘tar’ includes tar, tar.gz, and tar.bz files. The default ‘auto’ is [‘tar’, ‘zip’]. None or an empty list will return no matches found.
- **cache\_dir** – Location to store cached files, when None it defaults to the `[matchzoo.USER_DATA_DIR](~/.matchzoo/datasets)`.
- **verbose** – Verbosity mode, 0 (silent), 1 (verbose), 2 (semi-verbose)

**Papram extract** True tries extracting the file as an Archive, like tar or zip.

**Returns** Path to the downloaded file.



`matchzoo.utils._hash_file` (*fpath*, *algorithm*='sha256', *chunk\_size*=65535)  
Calculates a file sha256 or md5 hash.

#### Parameters

- **fpath** – path to the file being validated
- **algorithm** – hash algorithm, one of 'auto', 'sha256', or 'md5'. The default 'auto' detects the hash algorithm in use.
- **chunk\_size** – Bytes to read at a time, important for large files.

**Returns** The file hash.

## 3.1.2 Submodules

`matchzoo.version`

Matchzoo version file.

#### Module Contents

`matchzoo.version.__version__ = 0.0.1`

## 3.1.3 Package Contents

`matchzoo.USER_DIR`

`matchzoo.USER_DATA_DIR`

`matchzoo.USER_TUNED_MODELS_DIR`

`matchzoo.__version__ = 0.0.1`

**class** `matchzoo.DataPack` (*relation:pd.DataFrame*, *left:pd.DataFrame*, *right:pd.DataFrame*)

Bases: `object`

Matchzoo *DataPack* data structure, store dataframe and context.

*DataPack* is a MatchZoo native data structure that most MatchZoo data handling processes build upon. A *DataPack* consists of three parts: *left*, *right* and *relation*, each one of is a *pandas.DataFrame*.

#### Parameters

- **relation** – Store the relation between left document and right document use ids.
- **left** – Store the content or features for `id_left`.
- **right** – Store the content or features for `id_right`.

#### Example

```
>>> left = [
...     ['qid1', 'query 1'],
...     ['qid2', 'query 2']
... ]
>>> right = [
```

(continues on next page)

(continued from previous page)

```

...     ['did1', 'document 1'],
...     ['did2', 'document 2']
... ]
>>> relation = [['qid1', 'did1', 1], ['qid2', 'did2', 1]]
>>> relation_df = pd.DataFrame(relation)
>>> left = pd.DataFrame(left)
>>> right = pd.DataFrame(right)
>>> dp = DataPack(
...     relation=relation_df,
...     left=left,
...     right=right,
... )
>>> len(dp)
2

```

```
class FrameView(data_pack:'DataPack')
```

Bases: object

FrameView.

```
__getitem__(self, index:typing.Union[int, slice, np.array])
    Slicer.
```

```
__call__(self)
    Returns A full copy. Equivalent to frame[:].
```

```
DATA_FILENAME = data.dill
```

```
has_label
```

*True* if *label* column exists, *False* other wise.

**Type** return

```
frame
```

View the data pack as a `pandas.DataFrame`.

Returned data frame is created by merging the left data frame, the right dataframe and the relation data frame. Use `[]` to access an item or a slice of items.

**Returns** A `matchzoo.DataPack.FrameView` instance.

## Example

```

>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> type(data_pack.frame)
<class 'matchzoo.data_pack.data_pack.DataPack.FrameView'>
>>> frame_slice = data_pack.frame[0:5]
>>> type(frame_slice)
<class 'pandas.core.frame.DataFrame'>
>>> list(frame_slice.columns)
['id_left', 'text_left', 'id_right', 'text_right', 'label']
>>> full_frame = data_pack.frame()
>>> len(full_frame) == len(data_pack)
True

```

```
relation
```

*relation* getter.

**left**

Get `left()` of `DataPack`.

**right**

Get `right()` of `DataPack`.

**\_\_len\_\_**(*self*)

Get number of rows in the class:`DataPack` object.

**unpack**(*self*)

Unpack the data for training.

The return value can be directly feed to `model.fit` or `model.fit_generator`.

**Returns** A tuple of (X, y). y is *None* if *self* has no label.

**Example**

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> X, y = data_pack.unpack()
>>> type(X)
<class 'dict'>
>>> sorted(X.keys())
['id_left', 'id_right', 'text_left', 'text_right']
>>> type(y)
<class 'numpy.ndarray'>
>>> X, y = data_pack.drop_label().unpack()
>>> type(y)
<class 'NoneType'>
```

**\_\_getitem\_\_**(*self*, *index*:`typing.Union[int, slice, np.array]`)

Get specific item(s) as a new `DataPack`.

The returned `DataPack` will be a copy of the subset of the original `DataPack`.

**Parameters** *index* – Index of the item(s) to get.

**Returns** An instance of `DataPack`.

**copy**(*self*)

**Returns** A deep copy.

**save**(*self*, *dirpath*:`typing.Union[str, Path]`)

Save the `DataPack` object.

A saved `DataPack` is represented as a directory with a `DataPack` object (transformed user input as features and context), it will be saved by *pickle*.

**Parameters** *dirpath* – directory path of the saved `DataPack`.

**\_\_optional\_inplace**(*func*)

Decorator that adds *inplace* key word argument to a method.

Decorate any method that modifies *inplace* to make that *inplace* change optional.

**shuffle**(*self*)

Shuffle the data pack by shuffling the relation column.

**Parameters** *inplace* – *True* to modify *inplace*, *False* to return a modified copy. (default: *False*)

### Example

```
>>> import matchzoo as mz
>>> import numpy.random
>>> numpy.random.seed(0)
>>> data_pack = mz.datasets.toy.load_data()
>>> orig_ids = data_pack.relation['id_left']
>>> shuffled = data_pack.shuffle()
>>> (shuffled.relation['id_left'] != orig_ids).any()
True
```

#### **drop\_label**(self)

Remove *label* column from the data pack.

**Parameters** *inplace* – *True* to modify inplace, *False* to return a modified copy. (default: *False*)

### Example

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> data_pack.has_label
True
>>> data_pack.drop_label(inplace=True)
>>> data_pack.has_label
False
```

#### **append\_text\_length**(self, verbose=1)

Append *length\_left* and *length\_right* columns.

##### **Parameters**

- **inplace** – *True* to modify inplace, *False* to return a modified copy. (default: *False*)
- **verbose** – Verbosity.

### Example

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> 'length_left' in data_pack.frame[0].columns
False
>>> new_data_pack = data_pack.append_text_length(verbose=0)
>>> 'length_left' in new_data_pack.frame[0].columns
True
>>> 'length_left' in data_pack.frame[0].columns
False
>>> data_pack.append_text_length(inplace=True, verbose=0)
>>> 'length_left' in data_pack.frame[0].columns
True
```

#### **apply\_on\_text**(self, func:typing.Callable, mode:str='both', rename:typing.Optional[str]=None, verbose:int=1)

Apply *func* to text columns based on *mode*.

##### **Parameters**

- **func** – The function to apply.
- **mode** – One of “both”, “left” and “right”.
- **rename** – If set, use new names for results instead of replacing the original columns. To set *rename* in “both” mode, use a tuple of *str*, e.g. (“text\_left\_new\_name”, “text\_right\_new\_name”).
- **inplace** – *True* to modify inplace, *False* to return a modified copy. (default: *False*)
- **verbose** – Verbosity.

#### Examples::

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> frame = data_pack.frame
```

#### To apply *len* on the left text and add the result as ‘length\_left’:

```
>>> data_pack.apply_on_text(len, mode='left',
...                         rename='length_left',
...                         inplace=True,
...                         verbose=0)
>>> list(frame[0].columns) # noqa: E501
['id_left', 'text_left', 'length_left', 'id_right', 'text_right', 'label']
```

#### To do the same to the right text:

```
>>> data_pack.apply_on_text(len, mode='right',
...                         rename='length_right',
...                         inplace=True,
...                         verbose=0)
>>> list(frame[0].columns) # noqa: E501
['id_left', 'text_left', 'length_left', 'id_right', 'text_right', 'length_
↪right', 'label']
```

#### To do the same to the both texts at the same time:

```
>>> data_pack.apply_on_text(len, mode='both',
...                         rename=('extra_left', 'extra_right'),
...                         inplace=True,
...                         verbose=0)
>>> list(frame[0].columns) # noqa: E501
['id_left', 'text_left', 'length_left', 'extra_left', 'id_right', 'text_
↪right', 'length_right', 'extra_right', 'label']
```

#### To suppress outputs:

```
>>> data_pack.apply_on_text(len, mode='both', verbose=0,
...                         inplace=True)
```

`_apply_on_text_right` (*self*, *func*, *rename*, *verbose=1*)

`_apply_on_text_left` (*self*, *func*, *rename*, *verbose=1*)

`_apply_on_text_both` (*self*, *func*, *rename*, *verbose=1*)

`matchzoo.load_data_pack` (*dirpath:typing.Union[str, Path]*) → `DataPack`

Load a *DataPack*. The reverse function of `save()`.

**Parameters** `dirpath` – directory path of the saved model.

**Returns** a `DataPack` instance.

`matchzoo.chain_transform` (`units:typing.List[Unit]`)  $\rightarrow$  `typing.Callable`  
Compose unit transformations into a single function.

**Parameters** `units` – List of `matchzoo.StatelessUnit`.

`matchzoo.load_preprocessor` (`dirpath:typing.Union[str, Path]`)  $\rightarrow$  `'mz.DataPack'`  
Load the fitted `context`. The reverse function of `save()`.

**Parameters** `dirpath` – directory path of the saved model.

**Returns** a `DSSMPreprocessor` instance.

```
class matchzoo.Param (name:str, value:typing.Any=None, hyper_space:typing.Optional[SpaceType]=None,
                      validator:typing.Optional[typing.Callable[[typing.Any], bool]]=None,
                      desc:typing.Optional[str]=None)
```

Bases: `object`

Parameter class.

Basic usages with a name and value:

```
>>> param = Param('my_param', 10)
>>> param.name
'my_param'
>>> param.value
10
```

Use with a validator to make sure the parameter always keeps a valid value.

```
>>> param = Param(
...     name='my_param',
...     value=5,
...     validator=lambda x: 0 < x < 20
... )
>>> param.validator # doctest: +ELLIPSIS
<function <lambda> at 0x...>
>>> param.value
5
>>> param.value = 10
>>> param.value
10
>>> param.value = -1
Traceback (most recent call last):
...
ValueError: Validator not satisfied.
The validator's definition is as follows:
validator=lambda x: 0 < x < 20
```

Use with a hyper space. Setting up a hyper space for a parameter makes the parameter tunable in a `matchzoo.engine.Tuner`.

```
>>> from matchzoo.engine.hyper_spaces import quniform
>>> param = Param(
...     name='positive_num',
...     value=1,
...     hyper_space=quniform(low=1, high=5)
... )
```

(continues on next page)

(continued from previous page)

```
>>> param.hyper_space # doctest: +ELLIPSIS
<matchzoo.engine.hyper_spaces.quniform object at ...>
>>> from hyperopt.pyll.stochastic import sample
>>> hyperopt_space = param.hyper_space.convert(param.name)
>>> samples = [sample(hyperopt_space) for _ in range(64)]
>>> set(samples) == {1, 2, 3, 4, 5}
True
```

The boolean value of a *Param* instance is only *True* when the value is not *None*. This is because some default falsy values like zero or an empty list are valid parameter values. In other words, the boolean value means to be “if the parameter value is filled”.

```
>>> param = Param('dropout')
>>> if param:
...     print('OK')
>>> param = Param('dropout', 0)
>>> if param:
...     print('OK')
OK
```

A *\_pre\_assignment\_hook* is initialized as a data type convertor if the value is set as a number to keep data type consistency of the parameter. This conversion supports python built-in numbers, *numpy* numbers, and any number that inherits *numbers.Number*.

```
>>> param = Param('float_param', 0.5)
>>> param.value = 10
>>> param.value
10.0
>>> type(param.value)
<class 'float'>
```

**name**

Name of the parameter.

**Type** return**value**

Value of the parameter.

**Type** return**hyper\_space**

Hyper space of the parameter.

**Type** return**validator**

Validator of the parameter.

**Type** return**desc**

Parameter description.

**Type** return**\_infer\_pre\_assignment\_hook** (*self*)**\_validate** (*self*, *value*)**\_\_bool\_\_** (*self*)

**Returns** *False* when the value is *None*, *True* otherwise.

**set\_default** (*self*, *val*, *verbose=1*)

Set default value, has no effect if already has a value.

#### Parameters

- **val** – Default value to set.
- **verbose** – Verbosity.

**reset** (*self*)

Set the parameter's value to *None*, which means “not set”.

This method bypasses validator.

#### Example

```
>>> import matchzoo as mz
>>> param = mz.Param(
...     name='str', validator=lambda x: isinstance(x, str))
>>> param.value = 'hello'
>>> param.value = None
Traceback (most recent call last):
...
ValueError: Validator not satisfied.
The validator's definition is as follows:
name='str', validator=lambda x: isinstance(x, str))
>>> param.reset()
>>> param.value is None
True
```

**class** matchzoo.ParamTable

Bases: object

Parameter table class.

#### Example

```
>>> params = ParamTable()
>>> params.add(Param('ham', 'Parma Ham'))
>>> params.add(Param('egg', 'Over Easy'))
>>> params['ham']
'Parma Ham'
>>> params['egg']
'Over Easy'
>>> print(params)
ham                Parma Ham
egg                Over Easy
>>> params.add(Param('egg', 'Sunny side Up'))
Traceback (most recent call last):
...
ValueError: Parameter named egg already exists.
To re-assign parameter egg value, use `params["egg"] = value` instead.
```

**hyper\_space**

Hyper space of the table, a valid *hyperopt* graph.



**Type** return

**add** (*self*, *param*:*Param*)

**Parameters** *param* – parameter to add.

**get** (*self*, *key*)

**Returns** The parameter in the table named *key*.

**set** (*self*, *key*, *param*:*Param*)

Set *key* to parameter *param*.

**to\_frame** (*self*)

Convert the parameter table into a pandas data frame.

**Returns** A *pandas.DataFrame*.

### Example

```
>>> import matchzoo as mz
>>> table = mz.ParamTable()
>>> table.add(mz.Param(name='x', value=10, desc='my x'))
>>> table.add(mz.Param(name='y', value=20, desc='my y'))
>>> table.to_frame()
  Name Description  Value Hyper-Space
0    x      my x     10         None
1    y      my y     20         None
```

**\_\_getitem\_\_** (*self*, *key*:*str*)

**Returns** The value of the parameter in the table named *key*.

**\_\_setitem\_\_** (*self*, *key*:*str*, *value*:*typing.Any*)

Set the value of the parameter named *key*.

**Parameters**

- **key** – Name of the parameter.
- **value** – New value of the parameter to set.

**\_\_str\_\_** (*self*)

**Returns** Pretty formatted parameter table.

**\_\_iter\_\_** (*self*)

**Returns** A iterator that iterates over all parameter instances.

**completed** (*self*)

**Returns** *True* if all params are filled, *False* otherwise.

### Example

```
>>> import matchzoo
>>> model = matchzoo.models.DenseBaseline()
>>> model.params.completed()
False
```

**keys** (*self*)

**Returns** Parameter table keys.

**\_\_contains\_\_** (*self*, *item*)

**Returns** *True* if parameter in parameters.

**update** (*self*, *other:dict*)

Update *self*.

Update *self* with the key/value pairs from *other*, overwriting existing keys. Notice that this does not add new keys to *self*.

This method is usually used by models to obtain useful information from a preprocessor's context.

**Parameters** *other* – The dictionary used update.

### Example

```
>>> import matchzoo as mz
>>> model = mz.models.DenseBaseline()
>>> prpr = model.get_default_preprocessor()
>>> _ = prpr.fit(mz.datasets.toy.load_data(), verbose=0)
>>> model.params.update(prpr.context)
```

**class** matchzoo.**Embedding** (*data:dict*, *output\_dim:int*)

Bases: object

Embedding class.

**Examples::**

```
>>> import matchzoo as mz
>>> train_raw = mz.datasets.toy.load_data()
>>> pp = mz.preprocessors.NaivePreprocessor()
>>> train = pp.fit_transform(train_raw, verbose=0)
>>> vocab_unit = mz.build_vocab_unit(train, verbose=0)
>>> term_index = vocab_unit.state['term_index']
>>> embed_path = mz.datasets.embeddings.EMBED_RANK
```

**To load from a file:**

```
>>> embedding = mz.embedding.load_from_file(embed_path)
>>> matrix = embedding.build_matrix(term_index)
>>> matrix.shape[0] == len(term_index)
True
```

**To build your own:**

```
>>> data = {'A':[0, 1], 'B':[2, 3]}
>>> embedding = mz.Embedding(data, 2)
>>> matrix = embedding.build_matrix({'A': 2, 'B': 1, '_PAD': 0})
>>> matrix.shape == (3, 2)
True
```

**build\_matrix** (*self*, *term\_index:typing.Union[dict, mz.preprocessors.units.Vocabulary.TermIndex]*,  
*initializer=lambda: np.random.uniform(-0.2, 0.2))*

Build a matrix using *term\_index*.

**Parameters**

- **term\_index** – A *dict* or *TermIndex* to build with.

- **initializer** – A callable that returns a default value for missing terms in data. (default: a random uniform distribution in range)  $(-0.2, 0.2)$ ).

**Returns** A matrix.

```
matchzoo.build_unit_from_data_pack (unit:StatefulUnit,          data_pack:mz.DataPack,
                                   mode:str='both',  flatten:bool=True,  verbose:int=1)
                                   → StatefulUnit
```

Build a StatefulUnit from a *DataPack* object.

#### Parameters

- **unit** – StatefulUnit object to be built.
- **data\_pack** – The input *DataPack* object.
- **mode** – One of 'left', 'right', and 'both', to determine the source data for building the VocabularyUnit.
- **flatten** – Flatten the datapack or not. *True* to organize the *DataPack* text as a list, and *False* to organize *DataPack* text as a list of list.
- **verbose** – Verbosity.

**Returns** A built StatefulUnit object.

```
matchzoo.build_vocab_unit (data_pack:DataPack, mode:str='both', verbose:int=1) → Vocabulary
Build a preprocessor.units.Vocabulary given data_pack.
```

The *data\_pack* should be preprocessed beforehand, and each item in *text\_left* and *text\_right* columns of the *data\_pack* should be a list of tokens.

#### Parameters

- **data\_pack** – The *DataPack* to build vocabulary upon.
- **mode** – One of 'left', 'right', and 'both', to determine the source

data for building the VocabularyUnit. :param verbose: Verbosity. :return: A built vocabulary unit.



## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### m

matchzoo, 17

matchzoo.auto, 17

matchzoo.auto.preparer, 17

matchzoo.auto.preparer.prepare, 17

matchzoo.auto.preparer.preparer, 18

matchzoo.auto.tuner, 21

matchzoo.auto.tuner.tune, 21

matchzoo.auto.tuner.tuner, 22

matchzoo.data\_pack, 29

matchzoo.data\_pack.data\_pack, 29

matchzoo.data\_pack.pack, 34

matchzoo.dataloader, 40

matchzoo.dataloader.callbacks, 40

matchzoo.dataloader.callbacks.dynamic\_pooling, 40

matchzoo.dataloader.callbacks.histogram, 40

matchzoo.dataloader.callbacks.lambda\_callback, 41

matchzoo.dataloader.callbacks.padding, 42

matchzoo.dataloader.dataloader, 47

matchzoo.dataloader.dataloader\_builder, 48

matchzoo.dataloader.dataset, 49

matchzoo.dataloader.dataset\_builder, 51

matchzoo.dataloader.sampler, 51

matchzoo.datasets, 56

matchzoo.datasets.embeddings, 56

matchzoo.datasets.embeddings.load\_fasttext\_embedding, 56

matchzoo.datasets.embeddings.load\_glove\_embedding, 56

matchzoo.datasets.quora\_qp, 57

matchzoo.datasets.quora\_qp.load\_data, 57

matchzoo.datasets.snli, 58

matchzoo.datasets.snli.load\_data, 58

matchzoo.datasets.toy, 59

matchzoo.datasets.wiki\_qa, 60

matchzoo.datasets.wiki\_qa.load\_data, 60

matchzoo.embedding, 61

matchzoo.embedding.embedding, 61

matchzoo.engine, 63

matchzoo.engine.base\_callback, 63

matchzoo.engine.base\_metric, 63

matchzoo.engine.base\_model, 64

matchzoo.engine.base\_preprocessor, 66

matchzoo.engine.base\_task, 68

matchzoo.engine.hyper\_spaces, 68

matchzoo.engine.param, 71

matchzoo.engine.param\_table, 73

matchzoo.losses, 75

matchzoo.losses.rank\_cross\_entropy\_loss, 75

matchzoo.losses.rank\_hinge\_loss, 76

matchzoo.metrics, 78

matchzoo.metrics.accuracy, 78

matchzoo.metrics.average\_precision, 78

matchzoo.metrics.cross\_entropy, 79

matchzoo.metrics.discounted\_cumulative\_gain, 80

matchzoo.metrics.mean\_average\_precision, 80

matchzoo.metrics.mean\_reciprocal\_rank, 81

matchzoo.metrics.normalized\_discounted\_cumulative\_gain, 82

matchzoo.metrics.precision, 83

matchzoo.models, 87

matchzoo.models.arci, 87

matchzoo.models.arcii, 88

matchzoo.models.bert, 89

matchzoo.models.bimpm, 89

matchzoo.models.cdssm, 91

matchzoo.models.conv\_knrm, 92

matchzoo.models.dense\_baseline, 93

matchzoo.models.drmm, 93

`matchzoo.models.drmm_tks`, 94  
`matchzoo.models.dssm`, 95  
`matchzoo.models.esim`, 95  
`matchzoo.models.knrm`, 96  
`matchzoo.models.match_lstm`, 96  
`matchzoo.modules`, 104  
`matchzoo.modules.attention`, 104  
`matchzoo.modules.bert_module`, 105  
`matchzoo.modules.dropout`, 105  
`matchzoo.modules.gaussian_kernel`, 106  
`matchzoo.modules.matching`, 106  
`matchzoo.modules.stacked_brnn`, 107  
`matchzoo.preprocessors`, 110  
`matchzoo.preprocessors.basic_preprocessor`, 127  
`matchzoo.preprocessors.bert_preprocessor`, 128  
`matchzoo.preprocessors.build_unit_from_data_loader`, 129  
`matchzoo.preprocessors.build_vocab_unit`, 129  
`matchzoo.preprocessors.cdssm_preprocessor`, 130  
`matchzoo.preprocessors.chain_transform`, 130  
`matchzoo.preprocessors.diin_preprocessor`, 131  
`matchzoo.preprocessors.dssm_preprocessor`, 131  
`matchzoo.preprocessors.naive_preprocessor`, 132  
`matchzoo.preprocessors.units`, 110  
`matchzoo.preprocessors.units.character_index`, 111  
`matchzoo.preprocessors.units.digit_removal`, 111  
`matchzoo.preprocessors.units.frequency_filter`, 111  
`matchzoo.preprocessors.units.lemmatization`, 112  
`matchzoo.preprocessors.units.lowercase`, 113  
`matchzoo.preprocessors.units.matching_histogram`, 113  
`matchzoo.preprocessors.units.ngram_letter`, 114  
`matchzoo.preprocessors.units.punc_removal`, 114  
`matchzoo.preprocessors.units.stateful_unit`, 115  
`matchzoo.preprocessors.units.stemming`, 115  
`matchzoo.preprocessors.units.stop_removal`, 115  
`matchzoo.preprocessors.units.tokenize`, 116  
`matchzoo.preprocessors.units.truncated_length`, 116  
`matchzoo.preprocessors.units.unit`, 117  
`matchzoo.preprocessors.units.vocabulary`, 117  
`matchzoo.preprocessors.units.word_exact_match`, 118  
`matchzoo.preprocessors.units.word_hashing`, 119  
`matchzoo.tasks`, 137  
`matchzoo.tasks.classification`, 137  
`matchzoo.tasks.ranking`, 138  
`matchzoo.trainers`, 140  
`matchzoo.trainers.trainer`, 140  
`matchzoo.utils`, 146  
`matchzoo.utils.average_meter`, 146  
`matchzoo.utils.early_stopping`, 146  
`matchzoo.utils.get_file`, 147  
`matchzoo.utils.list_recursive_subclasses`, 149  
`matchzoo.utils.one_hot`, 149  
`matchzoo.utils.parse`, 149  
`matchzoo.utils.tensor_type`, 152  
`matchzoo.utils.timer`, 152  
`matchzoo.version`, 157



## Symbols

<code>__MATCH_PUNC</code>	(match- zoo.preprocessors.units.PuncRemoval attribute), 122	<code>__call__()</code>	(match- zoo.metrics.average_precision.AveragePrecision method), 79
<code>__MATCH_PUNC</code>	(match- zoo.preprocessors.units.punc_removal.PuncRemoval attribute), 114	<code>__call__()</code>	(match- zoo.metrics.cross_entropy.CrossEntropy method), 79
<code>__add__()</code>	(matchzoo.engine.hyper_spaces.HyperoptProxy method), 69	<code>__call__()</code>	(match- zoo.metrics.discounted_cumulative_gain.DiscountedCumulativeC method), 80
<code>__bool__()</code>	(matchzoo.Param method), 163	<code>__call__()</code>	(match- zoo.metrics.mean_average_precision.MeanAveragePrecision method), 81
<code>__bool__()</code>	(matchzoo.engine.param.Param method), 73	<code>__call__()</code>	(match- zoo.metrics.mean_reciprocal_rank.MeanReciprocalRank method), 81
<code>__call__()</code>	(matchzoo.DataPack.FrameView method), 158	<code>__call__()</code>	(match- zoo.metrics.normalized_discounted_cumulative_gain.Normalized method), 82
<code>__call__()</code>	(match- zoo.data_pack.DataPack.FrameView method), 35	<code>__call__()</code>	(matchzoo.metrics.precision.Precision method), 83
<code>__call__()</code>	(match- zoo.data_pack.data_pack.DataPack.FrameView method), 30	<code>__constants__</code>	(match- zoo.losses.RankCrossEntropyLoss attribute), 77
<code>__call__()</code>	(match- zoo.engine.base_metric.BaseMetric method), 64	<code>__constants__</code>	(matchzoo.losses.RankHingeLoss at- tribute), 77
<code>__call__()</code>	(matchzoo.metrics.Accuracy method), 86	<code>__constants__</code>	(match- zoo.losses.rank_cross_entropy_loss.RankCrossEntropyLoss attribute), 76
<code>__call__()</code>	(matchzoo.metrics.CrossEntropy method), 87	<code>__constants__</code>	(match- zoo.losses.rank_hinge_loss.RankHingeLoss attribute), 76
<code>__call__()</code>	(match- zoo.metrics.DiscountedCumulativeGain method), 84	<code>__contains__()</code>	(matchzoo.ParamTable method), 166
<code>__call__()</code>	(match- zoo.metrics.MeanAveragePrecision method), 85	<code>__contains__()</code>	(match- zoo.engine.param_table.ParamTable method), 75
<code>__call__()</code>	(matchzoo.metrics.MeanReciprocalRank method), 85	<code>__eq__()</code>	(matchzoo.engine.base_metric.BaseMetric method), 64
<code>__call__()</code>	(match- zoo.metrics.NormalizedDiscountedCumulativeGain method), 86	<code>__floordiv__()</code>	(match-
<code>__call__()</code>	(matchzoo.metrics.Precision method), 84		
<code>__call__()</code>	(matchzoo.metrics.accuracy.Accuracy		

`zoo.engine.hyper_spaces.HyperoptProxy`  
`method`), 70  
`__getitem__()` (`matchzoo.DataPack` `method`), 159  
`__getitem__()` (`matchzoo.DataPack.FrameView`  
`method`), 158  
`__getitem__()` (`matchzoo.ParamTable` `method`), 165  
`__getitem__()` (`matchzoo.data_pack.DataPack`  
`method`), 36  
`__getitem__()` (`match-`  
`zoo.data_pack.DataPack.FrameView` `method`),  
35  
`__getitem__()` (`match-`  
`zoo.data_pack.data_pack.DataPack` `method`),  
31  
`__getitem__()` (`match-`  
`zoo.data_pack.data_pack.DataPack.FrameView`  
`method`), 30  
`__getitem__()` (`matchzoo.dataloader.Dataset`  
`method`), 53  
`__getitem__()` (`match-`  
`zoo.dataloader.dataset.Dataset` `method`),  
50  
`__getitem__()` (`match-`  
`zoo.engine.param_table.ParamTable` `method`),  
74  
`__hash__()` (`match-`  
`zoo.engine.base_metric.BaseMetric` `method`),  
64  
`__iter__()` (`matchzoo.ParamTable` `method`), 165  
`__iter__()` (`matchzoo.dataloader.DataLoader`  
`method`), 55  
`__iter__()` (`match-`  
`zoo.dataloader.dataloader.DataLoader`  
`method`), 48  
`__iter__()` (`match-`  
`zoo.dataloader.sampler.BatchSampler` `method`),  
52  
`__iter__()` (`match-`  
`zoo.dataloader.sampler.RandomSampler`  
`method`), 52  
`__iter__()` (`match-`  
`zoo.dataloader.sampler.SequentialSampler`  
`method`), 51  
`__iter__()` (`match-`  
`zoo.dataloader.sampler.SortedSampler`  
`method`), 51  
`__iter__()` (`match-`  
`zoo.engine.param_table.ParamTable` `method`),  
75  
`__len__()` (`matchzoo.DataPack` `method`), 159  
`__len__()` (`matchzoo.data_pack.DataPack` `method`),  
36  
`__len__()` (`matchzoo.data_pack.data_pack.DataPack`  
`method`), 31  
`__len__()` (`matchzoo.dataloader.DataLoader`  
`method`), 54  
`__len__()` (`matchzoo.dataloader.Dataset` `method`), 53  
`__len__()` (`matchzoo.dataloader.dataloader.DataLoader`  
`method`), 48  
`__len__()` (`matchzoo.dataloader.dataset.Dataset`  
`method`), 50  
`__len__()` (`matchzoo.dataloader.sampler.BatchSampler`  
`method`), 52  
`__len__()` (`matchzoo.dataloader.sampler.RandomSampler`  
`method`), 52  
`__len__()` (`matchzoo.dataloader.sampler.SequentialSampler`  
`method`), 51  
`__len__()` (`matchzoo.dataloader.sampler.SortedSampler`  
`method`), 52  
`__missing__()` (`match-`  
`zoo.preprocessors.units.Vocabulary.TermIndex`  
`method`), 124  
`__missing__()` (`match-`  
`zoo.preprocessors.units.vocabulary.Vocabulary.TermIndex`  
`method`), 118  
`__mul__()` (`matchzoo.engine.hyper_spaces.HyperoptProxy`  
`method`), 69  
`__neg__()` (`matchzoo.engine.hyper_spaces.HyperoptProxy`  
`method`), 70  
`__pow__()` (`matchzoo.engine.hyper_spaces.HyperoptProxy`  
`method`), 70  
`__radd__()` (`match-`  
`zoo.engine.hyper_spaces.HyperoptProxy`  
`method`), 69  
`__repr__()` (`match-`  
`zoo.engine.base_metric.BaseMetric` `method`),  
64  
`__repr__()` (`matchzoo.metrics.Accuracy` `method`), 86  
`__repr__()` (`matchzoo.metrics.CrossEntropy`  
`method`), 87  
`__repr__()` (`match-`  
`zoo.metrics.DiscountedCumulativeGain`  
`method`), 84  
`__repr__()` (`match-`  
`zoo.metrics.MeanAveragePrecision` `method`),  
85  
`__repr__()` (`matchzoo.metrics.MeanReciprocalRank`  
`method`), 85  
`__repr__()` (`match-`  
`zoo.metrics.NormalizedDiscountedCumulativeGain`  
`method`), 86  
`__repr__()` (`matchzoo.metrics.Precision` `method`), 83  
`__repr__()` (`matchzoo.metrics.accuracy.Accuracy`  
`method`), 78  
`__repr__()` (`match-`  
`zoo.metrics.average_precision.AveragePrecision`  
`method`), 78  
`__repr__()` (`match-`

<code>zoo.metrics.cross_entropy.CrossEntropy</code> <code>method</code> ), 79	<code>zoo.engine.hyper_spaces.HyperoptProxy</code> <code>method</code> ), 70
<code>__repr__()</code> (match- <code>zoo.metrics.discounted_cumulative_gain.DiscountedCumulativeGain</code> <code>method</code> ), 80	<code>__version__</code> (in module <code>matchzoo</code> ), 157
<code>__repr__()</code> (match- <code>zoo.metrics.mean_average_precision.MeanAveragePrecision</code> <code>method</code> ), 81	<code>__apply_on_text_both__()</code> (matchzoo. <code>DataPack</code> <code>method</code> ), 161
<code>__repr__()</code> (match- <code>zoo.metrics.mean_reciprocal_rank.MeanReciprocalRank</code> <code>method</code> ), 81	<code>__apply_on_text_both__()</code> (match- <code>zoo.data_pack.DataPack</code> <code>method</code> ), 39
<code>__repr__()</code> (match- <code>zoo.metrics.normalized_discounted_cumulative_gain.NormalizedDiscountedCumulativeGain</code> <code>method</code> ), 82	<code>__apply_on_text_both__()</code> (match- <code>zoo.data_pack.data_pack.DataPack</code> <code>method</code> ), 34
<code>__repr__()</code> (matchzoo.metrics.precision.Precision <code>method</code> ), 83	<code>__apply_on_text_left__()</code> (matchzoo. <code>DataPack</code> <code>method</code> ), 39
<code>__rfloordiv__()</code> (match- <code>zoo.engine.hyper_spaces.HyperoptProxy</code> <code>method</code> ), 70	<code>__apply_on_text_left__()</code> (match- <code>zoo.data_pack.data_pack.DataPack</code> <code>method</code> ), 34
<code>__rmul__()</code> (match- <code>zoo.engine.hyper_spaces.HyperoptProxy</code> <code>method</code> ), 69	<code>__apply_on_text_right__()</code> (matchzoo. <code>DataPack</code> <code>method</code> ), 161
<code>__rpow__()</code> (match- <code>zoo.engine.hyper_spaces.HyperoptProxy</code> <code>method</code> ), 70	<code>__apply_on_text_right__()</code> (match- <code>zoo.data_pack.data_pack.DataPack</code> <code>method</code> ), 34
<code>__rsub__()</code> (match- <code>zoo.engine.hyper_spaces.HyperoptProxy</code> <code>method</code> ), 69	<code>__assure_losses__()</code> (match- <code>zoo.engine.base_task.BaseTask</code> <code>method</code> ), 68
<code>__rtruediv__()</code> (match- <code>zoo.engine.hyper_spaces.HyperoptProxy</code> <code>method</code> ), 70	<code>__assure_metrics__()</code> (match- <code>zoo.engine.base_task.BaseTask</code> <code>method</code> ), 68
<code>__setitem__()</code> (matchzoo. <code>ParamTable</code> <code>method</code> ), 165	<code>__backward__()</code> (matchzoo.trainers. <code>Trainer</code> <code>method</code> ), 144
<code>__setitem__()</code> (match- <code>zoo.engine.param_table.ParamTable</code> <code>method</code> ), 74	<code>__backward__()</code> (matchzoo.trainers.trainer. <code>Trainer</code> <code>method</code> ), 141
<code>__str__()</code> (matchzoo. <code>ParamTable</code> <code>method</code> ), 165	<code>__bfs__()</code> (in module <code>match-</code> <code>zoo.utils.list_recursive_subclasses</code> ), 149
<code>__str__()</code> (matchzoo.engine.hyper_spaces.choice <code>method</code> ), 70	<code>__build_dataloader_builder__()</code> (match- <code>zoo.auto.Preparer</code> <code>method</code> ), 28
<code>__str__()</code> (matchzoo.engine.hyper_spaces.quniform <code>method</code> ), 70	<code>__build_dataloader_builder__()</code> (match- <code>zoo.auto.preparer.Preparer</code> <code>method</code> ), 20
<code>__str__()</code> (matchzoo.engine.hyper_spaces.uniform <code>method</code> ), 70	<code>__build_dataloader_builder__()</code> (match- <code>zoo.auto.preparer.preparer.Preparer</code> <code>method</code> ), 19
<code>__str__()</code> (matchzoo.engine.param_table. <code>ParamTable</code> <code>method</code> ), 74	<code>__build_dataset_builder__()</code> (match- <code>zoo.auto.Preparer</code> <code>method</code> ), 28
<code>__str__()</code> (matchzoo.tasks.Classification <code>method</code> ), 139	<code>__build_dataset_builder__()</code> (match- <code>zoo.auto.preparer.Preparer</code> <code>method</code> ), 20
<code>__str__()</code> (matchzoo.tasks.Ranking <code>method</code> ), 140	<code>__build_dataset_builder__()</code> (match- <code>zoo.auto.preparer.preparer.Preparer</code> <code>method</code> ), 19
<code>__str__()</code> (matchzoo.tasks.classification.Classification <code>method</code> ), 137	<code>__build_match_histogram__()</code> (in module <code>match-</code> <code>zoo.dataloader.callbacks.histogram</code> ), 41
<code>__str__()</code> (matchzoo.tasks.ranking.Ranking <code>method</code> ), 138	<code>__build_matrix__()</code> (matchzoo.auto. <code>Preparer</code> <code>method</code> ),
<code>__sub__()</code> (matchzoo.engine.hyper_spaces.HyperoptProxy <code>method</code> ), 69	
<code>__truediv__()</code> (match-	

- [\\_build\\_matrix\(\)](#) (*matchzoo.auto.preparer.Preparer method*), 20
- [\\_build\\_matrix\(\)](#) (*matchzoo.auto.preparer.preparer.Preparer method*), 19
- [\\_build\\_model\(\)](#) (*matchzoo.auto.Preparer method*), 28
- [\\_build\\_model\(\)](#) (*matchzoo.auto.preparer.Preparer method*), 20
- [\\_build\\_model\(\)](#) (*matchzoo.auto.preparer.preparer.Preparer method*), 19
- [\\_convert\(\)](#) (*matchzoo.engine.base\_task.BaseTask method*), 68
- [\\_convert\\_to\\_list\\_index\(\)](#) (*in module matchzoo.data\_pack.data\_pack*), 29
- [\\_create\\_base\\_network\(\)](#) (*matchzoo.models.CDSSM method*), 99
- [\\_create\\_base\\_network\(\)](#) (*matchzoo.models.cdssm.CDSSM method*), 91
- [\\_create\\_full\\_params\(\)](#) (*matchzoo.auto.Tuner method*), 29
- [\\_create\\_full\\_params\(\)](#) (*matchzoo.auto.tuner.Tuner method*), 25
- [\\_create\\_full\\_params\(\)](#) (*matchzoo.auto.tuner.tuner.Tuner method*), 23
- [\\_default\\_units\(\)](#) (*matchzoo.engine.base\_preprocessor.BasePreprocessor class method*), 67
- [\\_default\\_units\(\)](#) (*matchzoo.preprocessors.CDSSMPreprocessor class method*), 136
- [\\_default\\_units\(\)](#) (*matchzoo.preprocessors.DSSMPreprocessor class method*), 133
- [\\_default\\_units\(\)](#) (*matchzoo.preprocessors.cdssm\_preprocessor.CDSSMPreprocessor class method*), 130
- [\\_default\\_units\(\)](#) (*matchzoo.preprocessors.dssm\_preprocessor.DSSMPreprocessor class method*), 132
- [\\_df\(\)](#) (*matchzoo.preprocessors.units.FrequencyFilter class method*), 121
- [\\_df\(\)](#) (*matchzoo.preprocessors.units.frequency\_filter.FrequencyFilter class method*), 112
- [\\_download\\_data\(\)](#) (*in module matchzoo.datasets.quora\_qp.load\_data*), 57
- [\\_download\\_data\(\)](#) (*in module matchzoo.datasets.snli.load\_data*), 58
- [\\_download\\_data\(\)](#) (*in module matchzoo.datasets.wiki\_qa.load\_data*), 60
- [\\_dynamic\\_pooling\\_index\(\)](#) (*in module matchzoo.dataloader.callbacks.dynamic\_pooling*), 40
- [\\_eval\\_metric\\_on\\_data\\_frame\(\)](#) (*matchzoo.trainers.Trainer class method*), 145
- [\\_eval\\_metric\\_on\\_data\\_frame\(\)](#) (*matchzoo.trainers.trainer.Trainer class method*), 142
- [\\_extract\\_archive\(\)](#) (*in module matchzoo.utils.get\_file*), 147
- [\\_fasttext\\_embedding\\_url](#) (*in module matchzoo.datasets.embeddings.load\_fasttext\_embedding*), 56
- [\\_filter\\_concrete\(\)](#) (*in module matchzoo.utils.list\_recursive\_subclasses*), 149
- [\\_fix\\_loss\\_sign\(\)](#) (*matchzoo.auto.Tuner method*), 29
- [\\_fix\\_loss\\_sign\(\)](#) (*matchzoo.auto.tuner.Tuner method*), 25
- [\\_fix\\_loss\\_sign\(\)](#) (*matchzoo.auto.tuner.tuner.Tuner method*), 24
- [\\_fmin\(\)](#) (*matchzoo.auto.Tuner method*), 29
- [\\_fmin\(\)](#) (*matchzoo.auto.tuner.Tuner method*), 25
- [\\_fmin\(\)](#) (*matchzoo.auto.tuner.tuner.Tuner method*), 23
- [\\_forward\\_unpadded\(\)](#) (*matchzoo.modules.StackedBRNN method*), 109
- [\\_forward\\_unpadded\(\)](#) (*matchzoo.modules.stacked\_brnn.StackedBRNN method*), 107
- [\\_gen\\_ids\(\)](#) (*in module matchzoo.data\_pack.pack*), 35
- [\\_glove\\_embedding\\_url](#) (*in module matchzoo.datasets.embeddings.load\_glove\_embedding*), 56
- [\\_handle\\_callbacks\\_on\\_batch\\_data\\_pack\(\)](#) (*matchzoo.dataloader.Dataset method*), 53
- [\\_handle\\_callbacks\\_on\\_batch\\_data\\_pack\(\)](#) (*matchzoo.dataloader.dataset.Dataset method*), 50
- [\\_handle\\_callbacks\\_on\\_batch\\_unpacked\(\)](#) (*matchzoo.dataloader.DataLoader method*), 55
- [\\_handle\\_callbacks\\_on\\_batch\\_unpacked\(\)](#) (*matchzoo.dataloader.Dataset method*), 53
- [\\_handle\\_callbacks\\_on\\_batch\\_unpacked\(\)](#) (*matchzoo.dataloader.dataloader.DataLoader method*), 48
- [\\_handle\\_callbacks\\_on\\_batch\\_unpacked\(\)](#) (*matchzoo.dataloader.dataset.Dataset method*), 50
- [\\_hash\\_file\(\)](#) (*in module matchzoo.utils*), 156
- [\\_hash\\_file\(\)](#) (*in module matchzoo.utils.get\_file*), 149
- [\\_idf\(\)](#) (*matchzoo.preprocessors.units.FrequencyFilter class method*), 121
- [\\_idf\(\)](#) (*matchzoo.preprocessors.units.frequency\_filter.FrequencyFilter class method*), 112
- [\\_infer\\_num\\_neg\(\)](#) (*matchzoo.auto.Preparer method*), 28

---

<code>_infer_num_neg()</code> ( <i>matchzoo.auto.preparer.Preparer method</i> ), 20	<code>_normalize_embedding()</code> ( <i>matchzoo.preprocessors.units.matching_histogram.MatchingHistogram method</i> ), 114
<code>_infer_num_neg()</code> ( <i>matchzoo.auto.preparer.preparer.Preparer method</i> ), 19	<code>_optional_inplace()</code> ( <i>matchzoo.DataPack method</i> ), 159
<code>_infer_pre_assignment_hook()</code> ( <i>matchzoo.Param method</i> ), 163	<code>_optional_inplace()</code> ( <i>matchzoo.data_pack.DataPack method</i> ), 37
<code>_infer_pre_assignment_hook()</code> ( <i>matchzoo.engine.param.Param method</i> ), 73	<code>_optional_inplace()</code> ( <i>matchzoo.data_pack.data_pack.DataPack method</i> ), 32
<code>_load_dataloader()</code> ( <i>matchzoo.trainers.Trainer method</i> ), 144	<code>_parse()</code> ( <i>in module matchzoo.utils.parse</i> ), 149
<code>_load_dataloader()</code> ( <i>matchzoo.trainers.trainer.Trainer method</i> ), 141	<code>_parse_metric()</code> ( <i>in module matchzoo.utils.parse</i> ), 151
<code>_load_model()</code> ( <i>matchzoo.trainers.Trainer method</i> ), 144	<code>_read_data()</code> ( <i>in module matchzoo.datasets.quora_qp.load_data</i> ), 57
<code>_load_model()</code> ( <i>matchzoo.trainers.trainer.Trainer method</i> ), 141	<code>_read_data()</code> ( <i>in module matchzoo.datasets.snli.load_data</i> ), 58
<code>_load_path()</code> ( <i>matchzoo.trainers.Trainer method</i> ), 144	<code>_read_data()</code> ( <i>in module matchzoo.datasets.wiki_qa.load_data</i> ), 60
<code>_load_path()</code> ( <i>matchzoo.trainers.trainer.Trainer method</i> ), 141	<code>_reorganize_pair_wise()</code> ( <i>matchzoo.dataloader.Dataset class method</i> ), 53
<code>_log_result()</code> ( <i>matchzoo.auto.Tuner class method</i> ), 29	<code>_reorganize_pair_wise()</code> ( <i>matchzoo.dataloader.dataset.Dataset class method</i> ), 50
<code>_log_result()</code> ( <i>matchzoo.auto.tuner.Tuner class method</i> ), 25	<code>_run()</code> ( <i>matchzoo.auto.Tuner method</i> ), 29
<code>_log_result()</code> ( <i>matchzoo.auto.tuner.tuner.Tuner class method</i> ), 24	<code>_run()</code> ( <i>matchzoo.auto.tuner.Tuner method</i> ), 25
<code>_make_conv_pool_block()</code> ( <i>matchzoo.models.ArcI class method</i> ), 103	<code>_run()</code> ( <i>matchzoo.auto.tuner.tuner.Tuner method</i> ), 23
<code>_make_conv_pool_block()</code> ( <i>matchzoo.models.ArcII class method</i> ), 103	<code>_run_epoch()</code> ( <i>matchzoo.trainers.Trainer method</i> ), 144
<code>_make_conv_pool_block()</code> ( <i>matchzoo.models.arci.ArcI class method</i> ), 88	<code>_run_epoch()</code> ( <i>matchzoo.trainers.trainer.Trainer method</i> ), 142
<code>_make_conv_pool_block()</code> ( <i>matchzoo.models.arcii.ArcII class method</i> ), 89	<code>_run_scheduler()</code> ( <i>matchzoo.trainers.Trainer method</i> ), 144
<code>_make_default_embedding_layer()</code> ( <i>matchzoo.engine.base_model.BaseModel method</i> ), 66	<code>_run_scheduler()</code> ( <i>matchzoo.trainers.trainer.Trainer method</i> ), 142
<code>_make_embedding_layer()</code> ( <i>matchzoo.engine.base_model.BaseModel method</i> ), 66	<code>_save()</code> ( <i>matchzoo.trainers.Trainer method</i> ), 145
<code>_make_multi_layer_perceptron_layer()</code> ( <i>matchzoo.engine.base_model.BaseModel method</i> ), 66	<code>_save()</code> ( <i>matchzoo.trainers.trainer.Trainer method</i> ), 142
<code>_make_output_layer()</code> ( <i>matchzoo.engine.base_model.BaseModel method</i> ), 66	<code>_set_param_default()</code> ( <i>matchzoo.engine.base_model.BaseModel method</i> ), 66
<code>_make_perceptron_layer()</code> ( <i>matchzoo.engine.base_model.BaseModel method</i> ), 66	<code>_tf()</code> ( <i>matchzoo.preprocessors.units.FrequencyFilter class method</i> ), 121
<code>_merge()</code> ( <i>in module matchzoo.data_pack.pack</i> ), 34	<code>_tf()</code> ( <i>matchzoo.preprocessors.units.frequency_filter.FrequencyFilter class method</i> ), 112
<code>_normalize_embedding()</code> ( <i>matchzoo.preprocessors.units.MatchingHistogram method</i> ), 122	<code>_trunc_text()</code> ( <i>in module matchzoo.dataloader.callbacks.histogram</i> ), 41
	<code>_url</code> ( <i>in module matchzoo.datasets.quora_qp.load_data</i> ), 57
	<code>_url</code> ( <i>in module matchzoo.datasets.snli.load_data</i> ), 58
	<code>_url</code> ( <i>in module matchzoo.datasets.wiki_qa.load_data</i> ), 60
	<code>_validate()</code> ( <i>matchzoo.Param method</i> ), 163
	<code>_validate()</code> ( <i>matchzoo.engine.param.Param</i>



method), 73

`_validate_dataloader()` (*matchzoo.auto.Tuner* class method), 29

`_validate_dataloader()` (*matchzoo.auto.tuner.Tuner* class method), 25

`_validate_dataloader()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 24

`_validate_kwargs()` (*matchzoo.auto.Tuner* class method), 29

`_validate_kwargs()` (*matchzoo.auto.tuner.Tuner* class method), 25

`_validate_kwargs()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 24

`_validate_matching_type()` (*matchzoo.modules.Matching* class method), 110

`_validate_matching_type()` (*matchzoo.modules.matching.Matching* class method), 107

`_validate_metric()` (*matchzoo.auto.Tuner* class method), 29

`_validate_metric()` (*matchzoo.auto.tuner.Tuner* class method), 25

`_validate_metric()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 24

`_validate_mode()` (*matchzoo.auto.Tuner* class method), 29

`_validate_mode()` (*matchzoo.auto.tuner.Tuner* class method), 25

`_validate_mode()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 24

`_validate_num_runs()` (*matchzoo.auto.Tuner* class method), 29

`_validate_num_runs()` (*matchzoo.auto.tuner.Tuner* class method), 25

`_validate_num_runs()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 24

`_validate_optimizer()` (*matchzoo.auto.Tuner* class method), 29

`_validate_optimizer()` (*matchzoo.auto.tuner.Tuner* class method), 25

`_validate_optimizer()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 24

`_validate_params()` (*matchzoo.auto.Tuner* class method), 29

`_validate_params()` (*matchzoo.auto.tuner.Tuner* class method), 25

`_validate_params()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 24

24

`_wrap_as_composite_func()` (in module *matchzoo.engine.hyper\_spaces*), 70

## A

Accuracy (class in *matchzoo.metrics*), 86

Accuracy (class in *matchzoo.metrics.accuracy*), 78

activation (in module *matchzoo.utils.parse*), 149

`add()` (*matchzoo.engine.param\_table.ParamTable* method), 74

`add()` (*matchzoo.ParamTable* method), 165

ALIAS (*matchzoo.engine.base\_metric.BaseMetric* attribute), 64

ALIAS (*matchzoo.engine.base\_metric.ClassificationMetric* attribute), 64

ALIAS (*matchzoo.engine.base\_metric.RankingMetric* attribute), 64

ALIAS (*matchzoo.metrics.Accuracy* attribute), 86

ALIAS (*matchzoo.metrics.accuracy.Accuracy* attribute), 78

ALIAS (*matchzoo.metrics.average\_precision.AveragePrecision* attribute), 78

ALIAS (*matchzoo.metrics.cross\_entropy.CrossEntropy* attribute), 79

ALIAS (*matchzoo.metrics.CrossEntropy* attribute), 87

ALIAS (*matchzoo.metrics.discounted\_cumulative\_gain.DiscountedCumulativeGain* attribute), 80

ALIAS (*matchzoo.metrics.DiscountedCumulativeGain* attribute), 84

ALIAS (*matchzoo.metrics.mean\_average\_precision.MeanAveragePrecision* attribute), 81

ALIAS (*matchzoo.metrics.mean\_reciprocal\_rank.MeanReciprocalRank* attribute), 81

ALIAS (*matchzoo.metrics.MeanAveragePrecision* attribute), 85

ALIAS (*matchzoo.metrics.MeanReciprocalRank* attribute), 85

ALIAS (*matchzoo.metrics.normalized\_discounted\_cumulative\_gain.NormalizedDiscountedCumulativeGain* attribute), 82

ALIAS (*matchzoo.metrics.NormalizedDiscountedCumulativeGain* attribute), 86

ALIAS (*matchzoo.metrics.Precision* attribute), 83

ALIAS (*matchzoo.metrics.precision.Precision* attribute), 83

`append_text_length()` (*matchzoo.data\_pack.data\_pack.DataPack* method), 32

`append_text_length()` (*matchzoo.data\_pack.DataPack* method), 37

`append_text_length()` (*matchzoo.DataPack* method), 160

`apply_on_text()` (*matchzoo.data\_pack.data\_pack.DataPack* method), 33

[apply\\_on\\_text\(\)](#) (*matchzoo.data\_pack.DataPack method*), 38  
[apply\\_on\\_text\(\)](#) (*matchzoo.DataPack method*), 160  
[ArcI](#) (*class in matchzoo.models*), 102  
[ArcI](#) (*class in matchzoo.models.arci*), 88  
[ArcII](#) (*class in matchzoo.models*), 103  
[ArcII](#) (*class in matchzoo.models.arci*), 88  
[Attention](#) (*class in matchzoo.modules*), 108  
[Attention](#) (*class in matchzoo.modules.attention*), 104  
[attention\(\)](#) (*in module matchzoo.models.bimpm*), 90  
[AverageMeter](#) (*class in matchzoo.utils*), 155  
[AverageMeter](#) (*class in matchzoo.utils.average\_meter*), 146  
[AveragePrecision](#) (*class in matchzoo.metrics.average\_precision*), 78  
[avg](#) (*matchzoo.utils.average\_meter.AverageMeter attribute*), 146  
[avg](#) (*matchzoo.utils.AverageMeter attribute*), 155

## B

[BaseCallback](#) (*class in matchzoo.engine.base\_callback*), 63  
[BaseMetric](#) (*class in matchzoo.engine.base\_metric*), 64  
[BaseModel](#) (*class in matchzoo.engine.base\_model*), 64  
[BasePreprocessor](#) (*class in matchzoo.engine.base\_preprocessor*), 67  
[BaseTask](#) (*class in matchzoo.engine.base\_task*), 68  
[BasicPadding](#) (*class in matchzoo.dataloader.callbacks*), 45  
[BasicPadding](#) (*class in matchzoo.dataloader.callbacks.padding*), 42  
[BasicPreprocessor](#) (*class in matchzoo.preprocessors*), 134  
[BasicPreprocessor](#) (*class in matchzoo.preprocessors.basic\_preprocessor*), 127  
[BatchSampler](#) (*class in matchzoo.dataloader.sampler*), 52  
[Bert](#) (*class in matchzoo.models*), 103  
[Bert](#) (*class in matchzoo.models.bert*), 89  
[BertModule](#) (*class in matchzoo.modules*), 110  
[BertModule](#) (*class in matchzoo.modules.bert\_module*), 105  
[BertPadding](#) (*class in matchzoo.dataloader.callbacks*), 46  
[BertPadding](#) (*class in matchzoo.dataloader.callbacks.padding*), 43  
[BertPreprocessor](#) (*class in matchzoo.preprocessors*), 136  
[BertPreprocessor](#) (*class in matchzoo.preprocessors.bert\_preprocessor*), 128  
[best\\_so\\_far](#) (*matchzoo.utils.early\_stopping.EarlyStopping attribute*), 146  
[best\\_so\\_far](#) (*matchzoo.utils.EarlyStopping attribute*), 155  
[BidirectionalAttention](#) (*class in matchzoo.modules*), 108  
[BidirectionalAttention](#) (*class in matchzoo.modules.attention*), 104  
[BiMPM](#) (*class in matchzoo.models*), 101  
[BiMPM](#) (*class in matchzoo.models.bimpm*), 89  
[build\(\)](#) (*matchzoo.dataloader.dataloader\_builder.DataLoaderBuilder method*), 49  
[build\(\)](#) (*matchzoo.dataloader.DataLoaderBuilder method*), 55  
[build\(\)](#) (*matchzoo.dataloader.dataset\_builder.DatasetBuilder method*), 51  
[build\(\)](#) (*matchzoo.dataloader.DatasetBuilder method*), 55  
[build\(\)](#) (*matchzoo.engine.base\_model.BaseModel method*), 66  
[build\(\)](#) (*matchzoo.models.ArcI method*), 103  
[build\(\)](#) (*matchzoo.models.arci.ArcI method*), 88  
[build\(\)](#) (*matchzoo.models.ArcII method*), 103  
[build\(\)](#) (*matchzoo.models.arci.ArcII method*), 89  
[build\(\)](#) (*matchzoo.models.Bert method*), 104  
[build\(\)](#) (*matchzoo.models.bert.Bert method*), 89  
[build\(\)](#) (*matchzoo.models.BiMPM method*), 101  
[build\(\)](#) (*matchzoo.models.bimpm.BiMPM method*), 90  
[build\(\)](#) (*matchzoo.models.CDSSM method*), 99  
[build\(\)](#) (*matchzoo.models.cdssm.CDSSM method*), 91  
[build\(\)](#) (*matchzoo.models.conv\_knrm.ConvKNRM method*), 92  
[build\(\)](#) (*matchzoo.models.ConvKNRM method*), 101  
[build\(\)](#) (*matchzoo.models.dense\_baseline.DenseBaseline method*), 93  
[build\(\)](#) (*matchzoo.models.DenseBaseline method*), 97  
[build\(\)](#) (*matchzoo.models.DRMM method*), 99  
[build\(\)](#) (*matchzoo.models.drmm.DRMM method*), 94  
[build\(\)](#) (*matchzoo.models.DRMMTKS method*), 100  
[build\(\)](#) (*matchzoo.models.drmmtk.DRMMTKS method*), 94  
[build\(\)](#) (*matchzoo.models.DSSM method*), 98  
[build\(\)](#) (*matchzoo.models.dssm.DSSM method*), 95  
[build\(\)](#) (*matchzoo.models.ESIM method*), 100  
[build\(\)](#) (*matchzoo.models.esim.ESIM method*), 96  
[build\(\)](#) (*matchzoo.models.KNRM method*), 101  
[build\(\)](#) (*matchzoo.models.knrm.KNRM method*), 96  
[build\(\)](#) (*matchzoo.models.MatchLSTM method*), 102  
[build\(\)](#) (*matchzoo.models.matchlstm.MatchLSTM method*), 97  
[build\\_matrix\(\)](#) (*matchzoo.Embedding method*), 166  
[build\\_matrix\(\)](#) (*matchzoo.embedding.Embedding method*), 62  
[build\\_matrix\(\)](#) (*matchzoo.embedding.embedding.Embedding method*), 62

method), 61  
 build\_unit\_from\_data\_pack() (in module matchzoo), 167  
 build\_unit\_from\_data\_pack() (in module matchzoo.preprocessors.build\_unit\_from\_data\_pack), 129  
 build\_vocab\_unit() (in module matchzoo), 167  
 build\_vocab\_unit() (in module matchzoo.preprocessors.build\_vocab\_unit), 129

## C

callbacks (matchzoo.dataloader.Dataset attribute), 53  
 callbacks (matchzoo.dataloader.dataset.Dataset attribute), 50  
 CDSSM (class in matchzoo.models), 98  
 CDSSM (class in matchzoo.models.cdssm), 91  
 CDSSMPadding (class in matchzoo.dataloader.callbacks), 46  
 CDSSMPadding (class in matchzoo.dataloader.callbacks.padding), 43  
 CDSSMPreprocessor (class in matchzoo.preprocessors), 135  
 CDSSMPreprocessor (class in matchzoo.preprocessors.cdssm\_preprocessor), 130  
 chain\_transform() (in module matchzoo), 162  
 chain\_transform() (in module matchzoo.preprocessors.chain\_transform), 131  
 CharacterIndex (class in matchzoo.preprocessors.units), 125  
 CharacterIndex (class in matchzoo.preprocessors.units.character\_index), 111  
 choice (class in matchzoo.engine.hyper\_spaces), 70  
 Classification (class in matchzoo.tasks), 138  
 Classification (class in matchzoo.tasks.classification), 137  
 ClassificationMetric (class in matchzoo.engine.base\_metric), 64  
 completed() (matchzoo.engine.param\_table.ParamTable method), 75  
 completed() (matchzoo.ParamTable method), 165  
 context (matchzoo.engine.base\_preprocessor.BasePreprocessor attribute), 67  
 context (matchzoo.preprocessors.units.stateful\_unit.StatefulUnit attribute), 115  
 context (matchzoo.preprocessors.units.StatefulUnit attribute), 123  
 convert() (matchzoo.engine.hyper\_spaces.HyperoptProxy method), 69  
 ConvKNRM (class in matchzoo.models), 101

ConvKNRM (class in matchzoo.models.conv\_knrm), 92  
 copy() (matchzoo.data\_pack.data\_pack.DataPack method), 32  
 copy() (matchzoo.data\_pack.DataPack method), 37  
 copy() (matchzoo.DataPack method), 159  
 CrossEntropy (class in matchzoo.metrics), 87  
 CrossEntropy (class in matchzoo.metrics.cross\_entropy), 79

## D

DATA\_FILENAME (matchzoo.data\_pack.data\_pack.DataPack attribute), 30  
 DATA\_FILENAME (matchzoo.data\_pack.DataPack attribute), 35  
 DATA\_FILENAME (matchzoo.DataPack attribute), 158  
 DATA\_FILENAME (matchzoo.engine.base\_preprocessor.BasePreprocessor attribute), 67  
 data\_pack (matchzoo.dataloader.Dataset attribute), 53  
 data\_pack (matchzoo.dataloader.dataset.Dataset attribute), 50  
 DATA\_ROOT (in module matchzoo.datasets.embeddings), 57  
 DataLoader (class in matchzoo.dataloader), 53  
 DataLoader (class in matchzoo.dataloader.dataloader), 47  
 DataLoaderBuilder (class in matchzoo.dataloader), 55  
 DataLoaderBuilder (class in matchzoo.dataloader.dataloader\_builder), 48  
 DataPack (class in matchzoo), 157  
 DataPack (class in matchzoo.data\_pack), 35  
 DataPack (class in matchzoo.data\_pack.data\_pack), 29  
 DataPack.FrameView (class in matchzoo), 158  
 DataPack.FrameView (class in matchzoo.data\_pack), 35  
 DataPack.FrameView (class in matchzoo.data\_pack.data\_pack), 30  
 Dataset (class in matchzoo.dataloader), 52  
 Dataset (class in matchzoo.dataloader.dataset), 49  
 DatasetBuilder (class in matchzoo.dataloader), 55  
 DatasetBuilder (class in matchzoo.dataloader.dataset\_builder), 51  
 DenseBaseline (class in matchzoo.models), 97  
 DenseBaseline (class in matchzoo.models.dense\_baseline), 93  
 desc (matchzoo.engine.param.Param attribute), 72  
 desc (matchzoo.Param attribute), 163  
 DigitRemoval (class in matchzoo.preprocessors.units), 120  
 DigitRemoval (class in matchzoo.preprocessors.units.digit\_removal), 111



DIINPadding (class in *matchzoo.dataloader.callbacks*), 46

DIINPadding (class in *matchzoo.dataloader.callbacks.padding*), 43

DIINPreprocessor (class in *matchzoo.preprocessors*), 136

DIINPreprocessor (class in *matchzoo.preprocessors.diin\_preprocessor*), 131

DiscountedCumulativeGain (class in *matchzoo.metrics*), 84

DiscountedCumulativeGain (class in *matchzoo.metrics.discounted\_cumulative\_gain*), 80

div\_with\_small\_value() (in module *matchzoo.models.bimpm*), 91

DRMM (class in *matchzoo.models*), 99

DRMM (class in *matchzoo.models.drmm*), 93

DRMMPadding (class in *matchzoo.dataloader.callbacks*), 45

DRMMPadding (class in *matchzoo.dataloader.callbacks.padding*), 42

DRMMTKS (class in *matchzoo.models*), 99

DRMMTKS (class in *matchzoo.models.drmmtk*), 94

drop\_label() (*matchzoo.data\_pack.data\_pack.DataPack* method), 32

drop\_label() (*matchzoo.data\_pack.DataPack* method), 37

drop\_label() (*matchzoo.DataPack* method), 160

dropout() (*matchzoo.models.BiMPM* method), 102

dropout() (*matchzoo.models.bimpm.BiMPM* method), 90

DSSM (class in *matchzoo.models*), 97

DSSM (class in *matchzoo.models.dssm*), 95

DSSMPreprocessor (class in *matchzoo.preprocessors*), 133

DSSMPreprocessor (class in *matchzoo.preprocessors.dssm\_preprocessor*), 131

DynamicPooling (class in *matchzoo.dataloader.callbacks*), 44

DynamicPooling (class in *matchzoo.dataloader.callbacks.dynamic\_pooling*), 40

## E

EarlyStopping (class in *matchzoo.utils*), 155

EarlyStopping (class in *matchzoo.utils.early\_stopping*), 146

EMBED\_10 (in module *matchzoo.datasets.embeddings*), 57

EMBED\_10\_GLOVE (in module *matchzoo.datasets.embeddings*), 57

EMBED\_RANK (in module *matchzoo.datasets.embeddings*), 57

Embedding (class in *matchzoo*), 166

Embedding (class in *matchzoo.embedding*), 62

Embedding (class in *matchzoo.embedding.embedding*), 61

ESIM (class in *matchzoo.models*), 100

ESIM (class in *matchzoo.models.esim*), 95

evaluate() (*matchzoo.trainers.Trainer* method), 145

evaluate() (*matchzoo.trainers.trainer.Trainer* method), 142

## F

fit() (*matchzoo.engine.base\_preprocessor.BasePreprocessor* method), 67

fit() (*matchzoo.preprocessors.basic\_preprocessor.BasicPreprocessor* method), 128

fit() (*matchzoo.preprocessors.BasicPreprocessor* method), 135

fit() (*matchzoo.preprocessors.bert\_preprocessor.BertPreprocessor* method), 128

fit() (*matchzoo.preprocessors.BertPreprocessor* method), 136

fit() (*matchzoo.preprocessors.cdssm\_preprocessor.CDSSMPreprocessor* method), 130

fit() (*matchzoo.preprocessors.CDSSMPreprocessor* method), 135

fit() (*matchzoo.preprocessors.diin\_preprocessor.DIINPreprocessor* method), 131

fit() (*matchzoo.preprocessors.DIINPreprocessor* method), 136

fit() (*matchzoo.preprocessors.dssm\_preprocessor.DSSMPreprocessor* method), 132

fit() (*matchzoo.preprocessors.DSSMPreprocessor* method), 133

fit() (*matchzoo.preprocessors.naive\_preprocessor.NaivePreprocessor* method), 132

fit() (*matchzoo.preprocessors.NaivePreprocessor* method), 134

fit() (*matchzoo.preprocessors.units.frequency\_filter.FrequencyFilter* method), 112

fit() (*matchzoo.preprocessors.units.FrequencyFilter* method), 121

fit() (*matchzoo.preprocessors.units.stateful\_unit.StatefulUnit* method), 115

fit() (*matchzoo.preprocessors.units.StatefulUnit* method), 123

fit() (*matchzoo.preprocessors.units.Vocabulary* method), 124

fit() (*matchzoo.preprocessors.units.vocabulary.Vocabulary* method), 118

fit\_kwargs (*matchzoo.auto.Tuner* attribute), 29

fit\_kwargs (*matchzoo.auto.tuner.Tuner* attribute), 25

fit\_kwargs (*matchzoo.auto.tuner.tuner.Tuner* attribute), 23

`fit_transform()` (*matchzoo.engine.base\_preprocessor.BasePreprocessor method*), 67  
`forward()` (*matchzoo.engine.base\_model.BaseModel method*), 66  
`forward()` (*matchzoo.losses.rank\_cross\_entropy\_loss.RankCrossEntropyLoss method*), 76  
`forward()` (*matchzoo.losses.rank\_hinge\_loss.RankHingeLoss method*), 76  
`forward()` (*matchzoo.losses.RankCrossEntropyLoss method*), 77  
`forward()` (*matchzoo.losses.RankHingeLoss method*), 77  
`forward()` (*matchzoo.models.ArcI method*), 103  
`forward()` (*matchzoo.models.arci.ArcI method*), 88  
`forward()` (*matchzoo.models.ArcII method*), 103  
`forward()` (*matchzoo.models.arcii.ArcII method*), 89  
`forward()` (*matchzoo.models.Bert method*), 104  
`forward()` (*matchzoo.models.bert.Bert method*), 89  
`forward()` (*matchzoo.models.BiMPM method*), 102  
`forward()` (*matchzoo.models.bimpm.BiMPM method*), 90  
`forward()` (*matchzoo.models.CDSSM method*), 99  
`forward()` (*matchzoo.models.cdssm.CDSSM method*), 92  
`forward()` (*matchzoo.models.cdssm.Squeeze method*), 92  
`forward()` (*matchzoo.models.conv\_knrm.ConvKNRM method*), 92  
`forward()` (*matchzoo.models.ConvKNRM method*), 101  
`forward()` (*matchzoo.models.dense\_baseline.DenseBaseline method*), 93  
`forward()` (*matchzoo.models.DenseBaseline method*), 97  
`forward()` (*matchzoo.models.DRMM method*), 99  
`forward()` (*matchzoo.models.drmm.DRMM method*), 94  
`forward()` (*matchzoo.models.DRMMTKS method*), 100  
`forward()` (*matchzoo.models.drmmtk.DRMMTKS method*), 94  
`forward()` (*matchzoo.models.DSSM method*), 98  
`forward()` (*matchzoo.models.dssm.DSSM method*), 95  
`forward()` (*matchzoo.models.ESIM method*), 100  
`forward()` (*matchzoo.models.esim.ESIM method*), 96  
`forward()` (*matchzoo.models.KNRM method*), 101  
`forward()` (*matchzoo.models.knrm.KNRM method*), 96  
`forward()` (*matchzoo.models.MatchLSTM method*), 102  
`forward()` (*matchzoo.models.matchlstm.MatchLSTM method*), 97  
`forward()` (*matchzoo.modules.Attention method*), 108  
`forward()` (*matchzoo.modules.attention.Attention method*), 104  
`forward()` (*matchzoo.modules.attention.BidirectionalAttention method*), 104  
`forward()` (*matchzoo.modules.attention.MatchModule method*), 105  
`forward()` (*matchzoo.modules.bert\_module.BertModule method*), 105  
`forward()` (*matchzoo.modules.BertModule method*), 110  
`forward()` (*matchzoo.modules.BidirectionalAttention method*), 108  
`forward()` (*matchzoo.modules.dropout.RNNDropout method*), 105  
`forward()` (*matchzoo.modules.gaussian\_kernel.GaussianKernel method*), 106  
`forward()` (*matchzoo.modules.GaussianKernel method*), 110  
`forward()` (*matchzoo.modules.Matching method*), 110  
`forward()` (*matchzoo.modules.matching.Matching method*), 107  
`forward()` (*matchzoo.modules.MatchModule method*), 108  
`forward()` (*matchzoo.modules.RNNDropout method*), 109  
`forward()` (*matchzoo.modules.stacked\_brnn.StackedBRNN method*), 107  
`forward()` (*matchzoo.modules.StackedBRNN method*), 109  
`frame` (*matchzoo.data\_pack.data\_pack.DataPack attribute*), 30  
`frame` (*matchzoo.data\_pack.DataPack attribute*), 35  
`frame` (*matchzoo.DataPack attribute*), 158  
`FrequencyFilter` (*class in matchzoo.preprocessors.units*), 120  
`FrequencyFilter` (*class in matchzoo.preprocessors.units.frequency\_filter*), 111

## G

`GaussianKernel` (*class in matchzoo.modules*), 109  
`GaussianKernel` (*class in matchzoo.modules.gaussian\_kernel*), 106  
`get()` (*matchzoo.engine.param\_table.ParamTable method*), 74  
`get()` (*matchzoo.ParamTable method*), 165  
`get_default_config()` (*matchzoo.auto.Preparer class method*), 28  
`get_default_config()` (*matchzoo.auto.preparer.Preparer class method*), 20  
`get_default_config()` (*matchzoo.auto.preparer.preparer.Preparer class method*), 19

<code>get_default_padding_callback()</code>	( <i>matchzoo.engine.base_model.BaseModel</i> class method), 66	<code>get_default_params()</code>	( <i>matchzoo.models.BiMPM</i> class method), 101
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.ArcI</i> class method), 103	<code>get_default_params()</code>	( <i>matchzoo.models.bimpm.BiMPM</i> class method), 90
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.arci.ArcI</i> class method), 88	<code>get_default_params()</code>	( <i>matchzoo.models.CDSSM</i> class method), 98
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.ArcII</i> class method), 103	<code>get_default_params()</code>	( <i>matchzoo.models.cdssm.CDSSM</i> class method), 91
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.arcii.ArcII</i> class method), 89	<code>get_default_params()</code>	( <i>matchzoo.models.conv_knrm.ConvKNRM</i> class method), 92
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.BiMPM</i> class method), 101	<code>get_default_params()</code>	( <i>matchzoo.models.ConvKNRM</i> class method), 101
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.bimpm.BiMPM</i> class method), 90	<code>get_default_params()</code>	( <i>matchzoo.models.dense_baseline.DenseBaseline</i> class method), 93
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.CDSSM</i> class method), 99	<code>get_default_params()</code>	( <i>matchzoo.models.DenseBaseline</i> class method), 97
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.cdssm.CDSSM</i> class method), 91	<code>get_default_params()</code>	( <i>matchzoo.models.DRMM</i> class method), 99
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.DRMM</i> class method), 99	<code>get_default_params()</code>	( <i>matchzoo.models.drmm.DRMM</i> class method), 94
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.drmm.DRMM</i> class method), 94	<code>get_default_params()</code>	( <i>matchzoo.models.DRMMTKS</i> class method), 100
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.DSSM</i> class method), 98	<code>get_default_params()</code>	( <i>matchzoo.models.drmmtk.DRMMTKS</i> class method), 94
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.dssm.DSSM</i> class method), 95	<code>get_default_params()</code>	( <i>matchzoo.models.DSSM</i> class method), 98
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.ESIM</i> class method), 100	<code>get_default_params()</code>	( <i>matchzoo.models.dssm.DSSM</i> class method), 95
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.esim.ESIM</i> class method), 96	<code>get_default_params()</code>	( <i>matchzoo.models.ESIM</i> class method), 100
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.MatchLSTM</i> class method), 102	<code>get_default_params()</code>	( <i>matchzoo.models.esim.ESIM</i> class method), 96
<code>get_default_padding_callback()</code>	( <i>matchzoo.models.matchlstm.MatchLSTM</i> class method), 97	<code>get_default_params()</code>	( <i>matchzoo.models.KNRM</i> class method), 100
<code>get_default_params()</code>	( <i>matchzoo.engine.base_model.BaseModel</i> class method), 65	<code>get_default_params()</code>	( <i>matchzoo.models.knrm.KNRM</i> class method), 96
<code>get_default_params()</code>	( <i>matchzoo.models.ArcI</i> class method), 103	<code>get_default_params()</code>	( <i>matchzoo.models.MatchLSTM</i> class method), 102
<code>get_default_params()</code>	( <i>matchzoo.models.arci.ArcI</i> class method), 88	<code>get_default_params()</code>	( <i>matchzoo.models.matchlstm.MatchLSTM</i> class method), 97
<code>get_default_params()</code>	( <i>matchzoo.models.ArcII</i> class method), 103	<code>get_default_preprocessor()</code>	( <i>matchzoo.engine.base_model.BaseModel</i> class method), 66
<code>get_default_params()</code>	( <i>matchzoo.models.arcii.ArcII</i> class method), 89	<code>get_default_preprocessor()</code>	( <i>matchzoo.models.bert.Bert</i> class method), 89
<code>get_default_params()</code>	( <i>matchzoo.models.Bert</i> class method), 103		
<code>get_default_params()</code>	( <i>matchzoo.models.bert.Bert</i> class method), 89		

`zoo.models.CDSSM` class method), 98  
`get_default_preprocessor()` (match-  
`zoo.models.cdssm.CDSSM` class method),  
 91  
`get_default_preprocessor()` (match-  
`zoo.models.DSSM` class method), 98  
`get_default_preprocessor()` (match-  
`zoo.models.dssm.DSSM` class method), 95  
`get_file()` (in module `matchzoo.utils`), 156  
`get_file()` (in module `matchzoo.utils.get_file`), 147  
`get_index_pool()` (`matchzoo.dataloader.Dataset`  
 method), 53  
`get_index_pool()` (match-  
`zoo.dataloader.dataset.Dataset` method),  
 50  
`guess_and_fill_missing_params()` (match-  
`zoo.engine.base_model.BaseModel` method),  
 66  
`guess_and_fill_missing_params()` (match-  
`zoo.models.CDSSM` method), 99  
`guess_and_fill_missing_params()` (match-  
`zoo.models.cdssm.CDSSM` method), 92

## H

`has_label` (`matchzoo.data_pack.data_pack.DataPack`  
 attribute), 30  
`has_label` (`matchzoo.data_pack.DataPack` attribute),  
 35  
`has_label` (`matchzoo.DataPack` attribute), 158  
`Histogram` (class in `matchzoo.dataloader.callbacks`),  
 45  
`Histogram` (class in `match-  
 zoo.dataloader.callbacks.histogram`), 41  
`hyper_space` (`matchzoo.engine.param.Param` at-  
 tribute), 72  
`hyper_space` (match-  
`zoo.engine.param_table.ParamTable` attribute),  
 74  
`hyper_space` (`matchzoo.Param` attribute), 163  
`hyper_space` (`matchzoo.ParamTable` attribute), 164  
`HyperoptProxy` (class in `match-  
 zoo.engine.hyper_spaces`), 69

## I

`id_left` (`matchzoo.dataloader.DataLoader` attribute),  
 54  
`id_left` (`matchzoo.dataloader.dataloader.DataLoader`  
 attribute), 48  
`index_pool` (`matchzoo.dataloader.Dataset` attribute),  
 53  
`index_pool` (`matchzoo.dataloader.dataset.Dataset` at-  
 tribute), 50  
`init_epoch()` (`matchzoo.dataloader.DataLoader`  
 method), 55

`init_epoch()` (match-  
`zoo.dataloader.dataloader.DataLoader`  
 method), 48  
`is_best_so_far` (match-  
`zoo.utils.early_stopping.EarlyStopping` at-  
 tribute), 147  
`is_best_so_far` (`matchzoo.utils.EarlyStopping` at-  
 tribute), 155

## K

`keys()` (`matchzoo.engine.param_table.ParamTable`  
 method), 75  
`keys()` (`matchzoo.ParamTable` method), 165  
`KNRM` (class in `matchzoo.models`), 100  
`KNRM` (class in `matchzoo.models.knrm`), 96

## L

`label` (`matchzoo.dataloader.DataLoader` attribute), 54  
`label` (`matchzoo.dataloader.dataloader.DataLoader` at-  
 tribute), 48  
`LambdaCallback` (class in `match-  
 zoo.dataloader.callbacks`), 44  
`LambdaCallback` (class in `match-  
 zoo.dataloader.callbacks.lambda_callback`),  
 41  
`left` (`matchzoo.data_pack.data_pack.DataPack` at-  
 tribute), 31  
`left` (`matchzoo.data_pack.DataPack` attribute), 36  
`left` (`matchzoo.DataPack` attribute), 158  
`Lemmatization` (class in `match-  
 zoo.preprocessors.units`), 121  
`Lemmatization` (class in `match-  
 zoo.preprocessors.units.lemmatization`), 112  
`list_available()` (in module `matchzoo.datasets`),  
 61  
`list_available()` (in module `matchzoo.metrics`),  
 87  
`list_available()` (in module `matchzoo.models`),  
 104  
`list_available()` (in module `match-  
 zoo.preprocessors`), 137  
`list_available()` (in module `match-  
 zoo.preprocessors.units`), 127  
`list_available_losses()` (match-  
`zoo.engine.base_task.BaseTask` class method),  
 68  
`list_available_losses()` (match-  
`zoo.tasks.Classification` class method), 139  
`list_available_losses()` (match-  
`zoo.tasks.classification.Classification` class  
 method), 137  
`list_available_losses()` (match-  
`zoo.tasks.Ranking` class method), 140

- `list_available_losses()` (*matchzoo.tasks.ranking.Ranking* class method), 138
- `list_available_metrics()` (*matchzoo.engine.base\_task.BaseTask* class method), 68
- `list_available_metrics()` (*matchzoo.tasks.Classification* class method), 139
- `list_available_metrics()` (*matchzoo.tasks.classification.Classification* class method), 137
- `list_available_metrics()` (*matchzoo.tasks.Ranking* class method), 140
- `list_available_metrics()` (*matchzoo.tasks.ranking.Ranking* class method), 138
- `list_recursive_concrete_subclasses()` (*in module matchzoo.utils*), 152
- `list_recursive_concrete_subclasses()` (*in module matchzoo.utils.list\_recursive\_subclasses*), 149
- `load_data()` (*in module matchzoo.datasets.quora\_qp*), 58
- `load_data()` (*in module matchzoo.datasets.quora\_qp.load\_data*), 57
- `load_data()` (*in module matchzoo.datasets.snli*), 59
- `load_data()` (*in module matchzoo.datasets.snli.load\_data*), 58
- `load_data()` (*in module matchzoo.datasets.toy*), 59
- `load_data()` (*in module matchzoo.datasets.wiki\_qa*), 60
- `load_data()` (*in module matchzoo.datasets.wiki\_qa.load\_data*), 60
- `load_data_pack()` (*in module matchzoo*), 161
- `load_data_pack()` (*in module matchzoo.data\_pack*), 39
- `load_data_pack()` (*in module matchzoo.data\_pack.data\_pack*), 34
- `load_embedding()` (*in module matchzoo.datasets.toy*), 59
- `load_fasttext_embedding()` (*in module matchzoo.datasets.embeddings*), 57
- `load_fasttext_embedding()` (*in module matchzoo.datasets.embeddings.load\_fasttext\_embedding*), 56
- `load_from_file()` (*in module matchzoo.embedding*), 62
- `load_from_file()` (*in module matchzoo.embedding.embedding*), 61
- `load_glove_embedding()` (*in module matchzoo.datasets.embeddings*), 57
- `load_glove_embedding()` (*in module matchzoo.datasets.embeddings.load\_glove\_embedding*), 56
- `load_preprocessor()` (*in module matchzoo*), 162
- `load_preprocessor()` (*in module matchzoo.engine.base\_preprocessor*), 67
- `load_state_dict()` (*matchzoo.utils.early\_stopping.EarlyStopping* method), 147
- `load_state_dict()` (*matchzoo.utils.early\_stopping.EarlyStopping* method), 156
- `loss` (*in module matchzoo.utils.parse*), 149
- `losses` (*matchzoo.engine.base\_task.BaseTask* attribute), 68
- `Lowercase` (*class in matchzoo.preprocessors.units*), 121
- `Lowercase` (*class in matchzoo.preprocessors.units.lowercase*), 113
- ## M
- `margin` (*matchzoo.losses.rank\_hinge\_loss.RankHingeLoss* attribute), 76
- `margin` (*matchzoo.losses.RankHingeLoss* attribute), 77
- `Matching` (*class in matchzoo.modules*), 110
- `Matching` (*class in matchzoo.modules.matching*), 106
- `MatchingHistogram` (*class in matchzoo.preprocessors.units*), 121
- `MatchingHistogram` (*class in matchzoo.preprocessors.units.matching\_histogram*), 113
- `MatchLSTM` (*class in matchzoo.models*), 102
- `MatchLSTM` (*class in matchzoo.models.matchlstm*), 97
- `MatchModule` (*class in matchzoo.modules*), 108
- `MatchModule` (*class in matchzoo.modules.attention*), 104
- `matchzoo` (*module*), 17
- `matchzoo.auto` (*module*), 17
- `matchzoo.auto.preparer` (*module*), 17
- `matchzoo.auto.preparer.prepare` (*module*), 17
- `matchzoo.auto.preparer.preparer` (*module*), 18
- `matchzoo.auto.tuner` (*module*), 21
- `matchzoo.auto.tuner.tune` (*module*), 21
- `matchzoo.auto.tuner.tuner` (*module*), 22
- `matchzoo.data_pack` (*module*), 29
- `matchzoo.data_pack.data_pack` (*module*), 29
- `matchzoo.data_pack.pack` (*module*), 34
- `matchzoo.data_loader` (*module*), 40
- `matchzoo.data_loader.callbacks` (*module*), 40
- `matchzoo.data_loader.callbacks.dynamic_pooling` (*module*), 40
- `matchzoo.data_loader.callbacks.histogram` (*module*), 40
- `matchzoo.data_loader.callbacks.lambda_callback` (*module*), 41



`matchzoo.dataloader.callbacks.padding` (module), 42

`matchzoo.dataloader.dataloader` (module), 47

`matchzoo.dataloader.dataloader_builder` (module), 48

`matchzoo.dataloader.dataset` (module), 49

`matchzoo.dataloader.dataset_builder` (module), 51

`matchzoo.dataloader.sampler` (module), 51

`matchzoo.datasets` (module), 56

`matchzoo.datasets.embeddings` (module), 56

`matchzoo.datasets.embeddings.load_fasttext_embedding` (module), 56

`matchzoo.datasets.embeddings.load_glove_embedding` (module), 56

`matchzoo.datasets.quora_qp` (module), 57

`matchzoo.datasets.quora_qp.load_data` (module), 57

`matchzoo.datasets.snli` (module), 58

`matchzoo.datasets.snli.load_data` (module), 58

`matchzoo.datasets.toy` (module), 59

`matchzoo.datasets.wiki_qa` (module), 60

`matchzoo.datasets.wiki_qa.load_data` (module), 60

`matchzoo.embedding` (module), 61

`matchzoo.embedding.embedding` (module), 61

`matchzoo.engine` (module), 63

`matchzoo.engine.base_callback` (module), 63

`matchzoo.engine.base_metric` (module), 63

`matchzoo.engine.base_model` (module), 64

`matchzoo.engine.base_preprocessor` (module), 66

`matchzoo.engine.base_task` (module), 68

`matchzoo.engine.hyper_spaces` (module), 68

`matchzoo.engine.param` (module), 71

`matchzoo.engine.param_table` (module), 73

`matchzoo.losses` (module), 75

`matchzoo.losses.rank_cross_entropy_loss` (module), 75

`matchzoo.losses.rank_hinge_loss` (module), 76

`matchzoo.metrics` (module), 78

`matchzoo.metrics.accuracy` (module), 78

`matchzoo.metrics.average_precision` (module), 78

`matchzoo.metrics.cross_entropy` (module), 79

`matchzoo.metrics.discounted_cumulative_gain` (module), 80

`matchzoo.metrics.mean_average_precision` (module), 80

`matchzoo.metrics.mean_reciprocal_rank` (module), 81

`matchzoo.metrics.normalized_discounted_cumulative_gain` (module), 82

`matchzoo.metrics.precision` (module), 83

`matchzoo.models` (module), 87

`matchzoo.models.arci` (module), 87

`matchzoo.models.arcii` (module), 88

`matchzoo.models.bert` (module), 89

`matchzoo.models.bimpm` (module), 89

`matchzoo.models.cdssm` (module), 91

`matchzoo.models.conv_knrm` (module), 92

`matchzoo.models.dense_baseline` (module), 93

`matchzoo.models.drmm` (module), 93

`matchzoo.models.drmm_tks` (module), 94

`matchzoo.models.dssm` (module), 95

`matchzoo.models.esim` (module), 95

`matchzoo.models.knrm` (module), 96

`matchzoo.models.matchlstm` (module), 96

`matchzoo.modules` (module), 104

`matchzoo.modules.attention` (module), 104

`matchzoo.modules.bert_module` (module), 105

`matchzoo.modules.dropout` (module), 105

`matchzoo.modules.gaussian_kernel` (module), 106

`matchzoo.modules.matching` (module), 106

`matchzoo.modules.stacked_brnn` (module), 107

`matchzoo.preprocessors` (module), 110

`matchzoo.preprocessors.basic_preprocessor` (module), 127

`matchzoo.preprocessors.bert_preprocessor` (module), 128

`matchzoo.preprocessors.build_unit_from_data_pack` (module), 129

`matchzoo.preprocessors.build_vocab_unit` (module), 129

`matchzoo.preprocessors.cdssm_preprocessor` (module), 130

`matchzoo.preprocessors.chain_transform` (module), 130

`matchzoo.preprocessors.diin_preprocessor` (module), 131

`matchzoo.preprocessors.dssm_preprocessor` (module), 131

`matchzoo.preprocessors.naive_preprocessor` (module), 132

`matchzoo.preprocessors.units` (module), 110

`matchzoo.preprocessors.units.character_index` (module), 111

`matchzoo.preprocessors.units.digit_removal` (module), 111

`matchzoo.preprocessors.units.frequency_filter` (module), 111

matchzoo.preprocessors.units.lemmatization (module), 112  
 matchzoo.preprocessors.units.lowercase (module), 113  
 matchzoo.preprocessors.units.matching\_histogram (module), 113  
 matchzoo.preprocessors.units.ngram\_letter (module), 114  
 matchzoo.preprocessors.units.punc\_removal (module), 114  
 matchzoo.preprocessors.units.stateful\_unit (module), 115  
 matchzoo.preprocessors.units.stemming (module), 115  
 matchzoo.preprocessors.units.stop\_removal (module), 115  
 matchzoo.preprocessors.units.tokenize (module), 116  
 matchzoo.preprocessors.units.truncated\_length (module), 116  
 matchzoo.preprocessors.units.unit (module), 117  
 matchzoo.preprocessors.units.vocabulary (module), 117  
 matchzoo.preprocessors.units.word\_exact\_match (module), 118  
 matchzoo.preprocessors.units.word\_hashing (module), 119  
 matchzoo.tasks (module), 137  
 matchzoo.tasks.classification (module), 137  
 matchzoo.tasks.ranking (module), 138  
 matchzoo.trainers (module), 140  
 matchzoo.trainers.trainer (module), 140  
 matchzoo.utils (module), 146  
 matchzoo.utils.average\_meter (module), 146  
 matchzoo.utils.early\_stopping (module), 146  
 matchzoo.utils.get\_file (module), 147  
 matchzoo.utils.list\_recursive\_subclasses (module), 149  
 matchzoo.utils.one\_hot (module), 149  
 matchzoo.utils.parse (module), 149  
 matchzoo.utils.tensor\_type (module), 152  
 matchzoo.utils.timer (module), 152  
 matchzoo.version (module), 157  
 MeanAveragePrecision (class in matchzoo.metrics), 85  
 MeanAveragePrecision (class in matchzoo.metrics.mean\_average\_precision), 81  
 MeanReciprocalRank (class in matchzoo.metrics), 85  
 MeanReciprocalRank (class in matchzoo.metrics.mean\_reciprocal\_rank), 81  
 metric (matchzoo.auto.Tuner attribute), 29  
 metric (matchzoo.auto.tuner.Tuner attribute), 25  
 metric (matchzoo.auto.tuner.tuner.Tuner attribute), 23  
 metrics (matchzoo.engine.base\_task.BaseTask attribute), 68  
 mode (matchzoo.auto.Tuner attribute), 29  
 mode (matchzoo.auto.tuner.Tuner attribute), 25  
 mode (matchzoo.auto.tuner.tuner.Tuner attribute), 23  
 mode (matchzoo.dataloader.Dataset attribute), 53  
 mode (matchzoo.dataloader.dataset.Dataset attribute), 50  
 mp\_matching\_func() (in module matchzoo.models.bimpm), 90  
 mp\_matching\_func\_pairwise() (in module matchzoo.models.bimpm), 90  
 mz\_collate() (in module matchzoo.dataloader.dataloader), 48  
**N**  
 NaivePreprocessor (class in matchzoo.preprocessors), 133  
 NaivePreprocessor (class in matchzoo.preprocessors.naive\_preprocessor), 132  
 name (matchzoo.engine.param.Param attribute), 72  
 name (matchzoo.Param attribute), 163  
 NgramLetter (class in matchzoo.preprocessors.units), 122  
 NgramLetter (class in matchzoo.preprocessors.units.ngram\_letter), 114  
 NormalizedDiscountedCumulativeGain (class in matchzoo.metrics), 86  
 NormalizedDiscountedCumulativeGain (class in matchzoo.metrics.normalized\_discounted\_cumulative\_gain), 82  
 num\_classes (matchzoo.tasks.Classification attribute), 139  
 num\_classes (matchzoo.tasks.classification.Classification attribute), 137  
 num\_dup (matchzoo.dataloader.Dataset attribute), 53  
 num\_dup (matchzoo.dataloader.dataset.Dataset attribute), 50  
 num\_neg (matchzoo.dataloader.Dataset attribute), 53  
 num\_neg (matchzoo.dataloader.dataset.Dataset attribute), 50  
 num\_neg (matchzoo.losses.rank\_cross\_entropy\_loss.RankCrossEntropyLoss attribute), 76  
 num\_neg (matchzoo.losses.rank\_hinge\_loss.RankHingeLoss attribute), 76  
 num\_neg (matchzoo.losses.RankCrossEntropyLoss attribute), 77  
 num\_neg (matchzoo.losses.RankHingeLoss attribute), 77

num\_runs (*matchzoo.auto.Tuner* attribute), 29  
 num\_runs (*matchzoo.auto.tuner.Tuner* attribute), 25  
 num\_runs (*matchzoo.auto.tuner.tuner.Tuner* attribute), 23

## O

on\_batch\_data\_pack () (*matchzoo.dataloader.callbacks.lambda\_callback.LambdaCallback* method), 42  
 on\_batch\_data\_pack () (*matchzoo.dataloader.callbacks.LambdaCallback* method), 44  
 on\_batch\_data\_pack () (*matchzoo.engine.base\_callback.BaseCallback* method), 63  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.BasicPadding* method), 45  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.BertPadding* method), 47  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.CDSSMPadding* method), 46  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.DIINPadding* method), 46  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.DRMPadding* method), 46  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.dynamic\_pooling.DynamicPooling* method), 40  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.DynamicPooling* method), 45  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.Histogram* method), 45  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.histogram.Histogram* method), 41  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.lambda\_callback.LambdaCallback* method), 42  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.LambdaCallback* method), 44  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.padding.BasicPadding* method), 42  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.padding.BertPadding* method), 44  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.padding.CDSSMPadding* method), 43  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.padding.DIINPadding* method), 43  
 on\_batch\_unpacked () (*matchzoo.dataloader.callbacks.padding.DRMPadding* method), 43  
 on\_batch\_unpacked () (*matchzoo.engine.base\_callback.BaseCallback* method), 63  
 one\_hot () (*in module matchzoo.utils*), 152  
 one\_hot () (*in module matchzoo.utils.one\_hot*), 149  
 optimizer (*in module matchzoo.utils.parse*), 149  
 output\_dtype (*matchzoo.engine.base\_task.BaseTask* attribute), 68  
 output\_dtype (*matchzoo.tasks.Classification* attribute), 139  
 output\_dtype (*matchzoo.tasks.classification.Classification* attribute), 137  
 output\_dtype (*matchzoo.tasks.Ranking* attribute), 140  
 output\_dtype (*matchzoo.tasks.ranking.Ranking* attribute), 138  
 output\_shape (*matchzoo.engine.base\_task.BaseTask* attribute), 68  
 output\_shape (*matchzoo.tasks.Classification* attribute), 139  
 output\_shape (*matchzoo.tasks.classification.Classification* attribute), 137  
 output\_shape (*matchzoo.tasks.Ranking* attribute), 139  
 output\_shape (*matchzoo.tasks.ranking.Ranking* attribute), 138

## P

pack () (*in module matchzoo.data\_pack*), 39  
 pack () (*in module matchzoo.data\_pack.pack*), 34  
 Param (*class in matchzoo*), 162  
 Param (*class in matchzoo.engine.param*), 71  
 param (*matchzoo.auto.Tuner* attribute), 28  
 params (*matchzoo.auto.tuner.Tuner* attribute), 24  
 params (*matchzoo.auto.tuner.tuner.Tuner* attribute), 23  
 params (*matchzoo.engine.base\_model.BaseModel* attribute), 65  
 ParamTable (*class in matchzoo*), 164  
 ParamTable (*class in matchzoo.engine.param\_table*), 73  
 parse\_activation () (*in module matchzoo.utils*), 153



[parse\\_activation\(\)](#) (in module `matchzoo.utils.parse`), 150  
[parse\\_loss\(\)](#) (in module `matchzoo.utils`), 153  
[parse\\_loss\(\)](#) (in module `matchzoo.utils.parse`), 150  
[parse\\_metric\(\)](#) (in module `matchzoo.utils`), 154  
[parse\\_metric\(\)](#) (in module `matchzoo.utils.parse`), 151  
[parse\\_optimizer\(\)](#) (in module `matchzoo.utils`), 154  
[parse\\_optimizer\(\)](#) (in module `matchzoo.utils.parse`), 151  
[Precision](#) (class in `matchzoo.metrics`), 83  
[Precision](#) (class in `matchzoo.metrics.precision`), 83  
[predict\(\)](#) (`matchzoo.trainers.Trainer` method), 145  
[predict\(\)](#) (`matchzoo.trainers.trainer.Trainer` method), 142  
[prepare\(\)](#) (in module `matchzoo.auto.preparer`), 20  
[prepare\(\)](#) (in module `matchzoo.auto.preparer.prepare`), 17  
[prepare\(\)](#) (`matchzoo.auto.Preparer` method), 27  
[prepare\(\)](#) (`matchzoo.auto.preparer.Preparer` method), 20  
[prepare\(\)](#) (`matchzoo.auto.preparer.preparer.Preparer` method), 19  
[Preparer](#) (class in `matchzoo.auto`), 27  
[Preparer](#) (class in `matchzoo.auto.preparer`), 19  
[Preparer](#) (class in `matchzoo.auto.preparer.preparer`), 18  
[Progbar](#) (class in `matchzoo.utils.get_file`), 147  
[PuncRemoval](#) (class in `matchzoo.preprocessors.units`), 122  
[PuncRemoval](#) (class in `matchzoo.preprocessors.units.punc_removal`), 114

## Q

[quniform](#) (class in `matchzoo.engine.hyper_spaces`), 70

## R

[RandomSampler](#) (class in `matchzoo.dataloader.sampler`), 52  
[RankCrossEntropyLoss](#) (class in `matchzoo.losses`), 77  
[RankCrossEntropyLoss](#) (class in `matchzoo.losses.rank_cross_entropy_loss`), 76  
[RankHingeLoss](#) (class in `matchzoo.losses`), 77  
[RankHingeLoss](#) (class in `matchzoo.losses.rank_hinge_loss`), 76  
[Ranking](#) (class in `matchzoo.tasks`), 139  
[Ranking](#) (class in `matchzoo.tasks.ranking`), 138  
[RankingMetric](#) (class in `matchzoo.engine.base_metric`), 64  
[relation](#) (`matchzoo.data_pack.data_pack.DataPack` attribute), 31  
[relation](#) (`matchzoo.data_pack.DataPack` attribute), 36

[relation](#) (`matchzoo.DataPack` attribute), 158  
[reset\(\)](#) (`matchzoo.engine.param.Param` method), 73  
[reset\(\)](#) (`matchzoo.Param` method), 164  
[reset\(\)](#) (`matchzoo.utils.average_meter.AverageMeter` method), 146  
[reset\(\)](#) (`matchzoo.utils.AverageMeter` method), 155  
[reset\(\)](#) (`matchzoo.utils.Timer` method), 155  
[reset\(\)](#) (`matchzoo.utils.timer.Timer` method), 152  
[reset\\_parameters\(\)](#) (`matchzoo.models.BiMPM` method), 102  
[reset\\_parameters\(\)](#) (`matchzoo.models.bimpm.BiMPM` method), 90  
[restore\(\)](#) (`matchzoo.trainers.Trainer` method), 145  
[restore\(\)](#) (`matchzoo.trainers.trainer.Trainer` method), 143  
[restore\\_model\(\)](#) (`matchzoo.trainers.Trainer` method), 145  
[restore\\_model\(\)](#) (`matchzoo.trainers.trainer.Trainer` method), 143  
[resume\(\)](#) (`matchzoo.utils.Timer` method), 155  
[resume\(\)](#) (`matchzoo.utils.timer.Timer` method), 152  
[right](#) (`matchzoo.data_pack.data_pack.DataPack` attribute), 31  
[right](#) (`matchzoo.data_pack.DataPack` attribute), 36  
[right](#) (`matchzoo.DataPack` attribute), 159  
[RNNDropout](#) (class in `matchzoo.modules`), 108  
[RNNDropout](#) (class in `matchzoo.modules.dropout`), 105  
[run\(\)](#) (`matchzoo.trainers.Trainer` method), 144  
[run\(\)](#) (`matchzoo.trainers.trainer.Trainer` method), 142

## S

[sample\(\)](#) (in module `matchzoo.engine.hyper_spaces`), 70  
[sample\(\)](#) (`matchzoo.dataloader.Dataset` method), 53  
[sample\(\)](#) (`matchzoo.dataloader.dataset.Dataset` method), 50  
[save\(\)](#) (`matchzoo.data_pack.data_pack.DataPack` method), 32  
[save\(\)](#) (`matchzoo.data_pack.DataPack` method), 37  
[save\(\)](#) (`matchzoo.DataPack` method), 159  
[save\(\)](#) (`matchzoo.engine.base_preprocessor.BasePreprocessor` method), 67  
[save\(\)](#) (`matchzoo.trainers.Trainer` method), 145  
[save\(\)](#) (`matchzoo.trainers.trainer.Trainer` method), 142  
[save\\_model\(\)](#) (`matchzoo.trainers.Trainer` method), 145  
[save\\_model\(\)](#) (`matchzoo.trainers.trainer.Trainer` method), 142  
[SequentialSampler](#) (class in `matchzoo.dataloader.sampler`), 51  
[set\(\)](#) (`matchzoo.engine.param_table.ParamTable` method), 74  
[set\(\)](#) (`matchzoo.ParamTable` method), 165

- [set\\_default\(\)](#) (*matchzoo.engine.param.Param method*), 73  
[set\\_default\(\)](#) (*matchzoo.Param method*), 164  
[should\\_stop\\_early](#) (*matchzoo.utils.early\_stopping.EarlyStopping attribute*), 147  
[should\\_stop\\_early](#) (*matchzoo.utils.EarlyStopping attribute*), 156  
[shuffle\(\)](#) (*matchzoo.data\_pack.data\_pack.DataPack method*), 32  
[shuffle\(\)](#) (*matchzoo.data\_pack.DataPack method*), 37  
[shuffle\(\)](#) (*matchzoo.dataloader.Dataset method*), 53  
[shuffle\(\)](#) (*matchzoo.dataloader.dataset.Dataset method*), 50  
[shuffle\(\)](#) (*matchzoo.DataPack method*), 159  
[sort\(\)](#) (*matchzoo.dataloader.Dataset method*), 53  
[sort\(\)](#) (*matchzoo.dataloader.dataset.Dataset method*), 50  
[sort\\_and\\_couple\(\)](#) (*in module matchzoo.engine.base\_metric*), 64  
[SortedSampler](#) (*class in matchzoo.dataloader.sampler*), 51  
[SpaceType](#) (*in module matchzoo.engine.param*), 71  
[Squeeze](#) (*class in matchzoo.models.cdssm*), 92  
[StackedBRNN](#) (*class in matchzoo.modules*), 109  
[StackedBRNN](#) (*class in matchzoo.modules.stacked\_brnn*), 107  
[state](#) (*matchzoo.preprocessors.units.stateful\_unit.StatefulUnit attribute*), 115  
[state](#) (*matchzoo.preprocessors.units.StatefulUnit attribute*), 123  
[state\\_dict\(\)](#) (*matchzoo.utils.early\_stopping.EarlyStopping method*), 147  
[state\\_dict\(\)](#) (*matchzoo.utils.EarlyStopping method*), 156  
[StatefulUnit](#) (*class in matchzoo.preprocessors.units*), 122  
[StatefulUnit](#) (*class in matchzoo.preprocessors.units.stateful\_unit*), 115  
[Stemming](#) (*class in matchzoo.preprocessors.units*), 123  
[Stemming](#) (*class in matchzoo.preprocessors.units.stemming*), 115  
[stop\(\)](#) (*matchzoo.utils.Timer method*), 155  
[stop\(\)](#) (*matchzoo.utils.timer.Timer method*), 152  
[StopRemoval](#) (*class in matchzoo.preprocessors.units*), 123  
[StopRemoval](#) (*class in matchzoo.preprocessors.units.stop\_removal*), 115  
[stopwords](#) (*matchzoo.preprocessors.units.stop\_removal.StopRemoval attribute*), 116  
[stopwords](#) (*matchzoo.preprocessors.units.StopRemoval attribute*), 123
- T**  
[TensorType](#) (*in module matchzoo.utils*), 152  
[TensorType](#) (*in module matchzoo.utils.tensor\_type*), 152  
[time](#) (*matchzoo.utils.Timer attribute*), 155  
[time](#) (*matchzoo.utils.timer.Timer attribute*), 152  
[Timer](#) (*class in matchzoo.utils*), 155  
[Timer](#) (*class in matchzoo.utils.timer*), 152  
[to\\_frame\(\)](#) (*matchzoo.engine.param\_table.ParamTable method*), 74  
[to\\_frame\(\)](#) (*matchzoo.ParamTable method*), 165  
[Tokenize](#) (*class in matchzoo.preprocessors.units*), 123  
[Tokenize](#) (*class in matchzoo.preprocessors.units.tokenize*), 116  
[Trainer](#) (*class in matchzoo.trainers*), 143  
[Trainer](#) (*class in matchzoo.trainers.trainer*), 140  
[trainloader](#) (*matchzoo.auto.Tuner attribute*), 28  
[trainloader](#) (*matchzoo.auto.tuner.Tuner attribute*), 25  
[trainloader](#) (*matchzoo.auto.tuner.tuner.Tuner attribute*), 23  
[transform\(\)](#) (*matchzoo.engine.base\_preprocessor.BasePreprocessor method*), 67  
[transform\(\)](#) (*matchzoo.preprocessors.basic\_preprocessor.BasicPreprocessor method*), 128  
[transform\(\)](#) (*matchzoo.preprocessors.BasicPreprocessor method*), 135  
[transform\(\)](#) (*matchzoo.preprocessors.bert\_preprocessor.BertPreprocessor method*), 128  
[transform\(\)](#) (*matchzoo.preprocessors.BertPreprocessor method*), 136  
[transform\(\)](#) (*matchzoo.preprocessors.cdssm\_preprocessor.CDSSMPreprocessor method*), 130  
[transform\(\)](#) (*matchzoo.preprocessors.CDSSMPreprocessor method*), 136  
[transform\(\)](#) (*matchzoo.preprocessors.diin\_preprocessor.DIINPreprocessor method*), 131  
[transform\(\)](#) (*matchzoo.preprocessors.DIINPreprocessor method*), 136  
[transform\(\)](#) (*matchzoo.preprocessors.dssm\_preprocessor.DSSMPreprocessor method*), 132  
[transform\(\)](#) (*matchzoo.preprocessors.DSSMPreprocessor*

`method)`, 133  
`transform()` (`match-zoo.preprocessors.naive_preprocessor.NaivePreprocessor` `method)`, 133  
`transform()` (`match-zoo.preprocessors.NaivePreprocessor` `method)`, 134  
`transform()` (`match-zoo.preprocessors.units.character_index.CharacterIndex` `method)`, 111  
`transform()` (`match-zoo.preprocessors.units.CharacterIndex` `method)`, 125  
`transform()` (`match-zoo.preprocessors.units.digit_removal.DigitRemoval` `method)`, 111  
`transform()` (`match-zoo.preprocessors.units.DigitRemoval` `method)`, 120  
`transform()` (`match-zoo.preprocessors.units.frequency_filter.FrequencyFilter` `method)`, 112  
`transform()` (`match-zoo.preprocessors.units.FrequencyFilter` `method)`, 121  
`transform()` (`match-zoo.preprocessors.units.Lemmatization` `method)`, 121  
`transform()` (`match-zoo.preprocessors.units.lemmatization.Lemmatization` `method)`, 112  
`transform()` (`match-zoo.preprocessors.units.Lowercase` `method)`, 121  
`transform()` (`match-zoo.preprocessors.units.lowercase.Lowercase` `method)`, 113  
`transform()` (`match-zoo.preprocessors.units.matching_histogram.MatchingHistogram` `method)`, 114  
`transform()` (`match-zoo.preprocessors.units.MatchingHistogram` `method)`, 122  
`transform()` (`match-zoo.preprocessors.units.ngram_letter.NgramLetter` `method)`, 114  
`transform()` (`match-zoo.preprocessors.units.NgramLetter` `method)`, 122  
`transform()` (`match-zoo.preprocessors.units.punc_removal.PuncRemoval` `method)`, 115  
`transform()` (`match-zoo.preprocessors.units.PuncRemoval` `method)`, 122  
`transform()` (`match-zoo.preprocessors.units.Stemming` `method)`, 123  
`transform()` (`match-zoo.preprocessors.units.stemming.Stemming` `method)`, 115  
`transform()` (`match-zoo.preprocessors.units.stop_removal.StopRemoval` `method)`, 116  
`transform()` (`match-zoo.preprocessors.units.StopRemoval` `method)`, 123  
`transform()` (`matchzoo.preprocessors.units.Tokenize` `method)`, 123  
`transform()` (`matchzoo.preprocessors.units.tokenize.Tokenize` `method)`, 116  
`transform()` (`matchzoo.preprocessors.units.truncated_length.TruncatedLength` `method)`, 117  
`transform()` (`matchzoo.preprocessors.units.TruncatedLength` `method)`, 126  
`transform()` (`matchzoo.preprocessors.units.Unit` `method)`, 120  
`transform()` (`matchzoo.preprocessors.units.unit.Unit` `method)`, 117  
`transform()` (`match-zoo.preprocessors.units.Vocabulary` `method)`, 124  
`transform()` (`match-zoo.preprocessors.units.vocabulary.Vocabulary` `method)`, 118  
`transform()` (`match-zoo.preprocessors.units.word_exact_match.WordExactMatch` `method)`, 119  
`transform()` (`match-zoo.preprocessors.units.word_hashing.WordHashing` `method)`, 119  
`transform()` (`match-zoo.preprocessors.units.WordExactMatch` `method)`, 126  
`transform()` (`match-zoo.preprocessors.units.WordHashing` `method)`, 125  
`TruncatedLength` (class in `match-zoo.preprocessors.units`), 126  
`TruncatedLength` (class in `match-zoo.preprocessors.units.truncated_length`), 116  
`tune()` (in module `matchzoo.auto.tuner`), 25  
`tune()` (in module `matchzoo.auto.tuner.tune`), 21  
`tune()` (`matchzoo.auto.Tuner` `method)`, 29

`tune()` (*matchzoo.auto.tuner.Tuner* method), 25  
`tune()` (*matchzoo.auto.tuner.tuner.Tuner* method), 23  
`Tuner` (class in *matchzoo.auto*), 28  
`Tuner` (class in *matchzoo.auto.tuner*), 24  
`Tuner` (class in *matchzoo.auto.tuner.tuner*), 22  
`TYPE` (*matchzoo.engine.base\_task.BaseTask* attribute), 68  
`TYPE` (*matchzoo.tasks.Classification* attribute), 139  
`TYPE` (*matchzoo.tasks.classification.Classification* attribute), 137  
`TYPE` (*matchzoo.tasks.Ranking* attribute), 139  
`TYPE` (*matchzoo.tasks.ranking.Ranking* attribute), 138

## U

`uniform` (class in *matchzoo.engine.hyper\_spaces*), 70  
`Unit` (class in *matchzoo.preprocessors.units*), 120  
`Unit` (class in *matchzoo.preprocessors.units.unit*), 117  
`unpack()` (*matchzoo.data\_pack.data\_pack.DataPack* method), 31  
`unpack()` (*matchzoo.data\_pack.DataPack* method), 36  
`unpack()` (*matchzoo.DataPack* method), 159  
`update()` (*matchzoo.engine.param\_table.ParamTable* method), 75  
`update()` (*matchzoo.ParamTable* method), 166  
`update()` (*matchzoo.utils.average\_meter.AverageMeter* method), 146  
`update()` (*matchzoo.utils.AverageMeter* method), 155  
`update()` (*matchzoo.utils.early\_stopping.EarlyStopping* method), 147  
`update()` (*matchzoo.utils.EarlyStopping* method), 156  
`update()` (*matchzoo.utils.get\_file.Progbar* method), 147  
`USER_DATA_DIR` (in module *matchzoo*), 157  
`USER_DIR` (in module *matchzoo*), 157  
`USER_TUNED_MODELS_DIR` (in module *matchzoo*), 157

## V

`validate_context()` (in module *matchzoo.engine.base\_preprocessor*), 67  
`validate_file()` (in module *matchzoo.utils.get\_file*), 148  
`validator` (*matchzoo.engine.param.Param* attribute), 72  
`validator` (*matchzoo.Param* attribute), 163  
`validloader` (*matchzoo.auto.Tuner* attribute), 28  
`validloader` (*matchzoo.auto.tuner.Tuner* attribute), 25  
`validloader` (*matchzoo.auto.tuner.tuner.Tuner* attribute), 23  
`value` (*matchzoo.engine.param.Param* attribute), 72  
`value` (*matchzoo.Param* attribute), 163  
`verbose` (*matchzoo.auto.Tuner* attribute), 29  
`verbose` (*matchzoo.auto.tuner.Tuner* attribute), 25

`verbose` (*matchzoo.auto.tuner.tuner.Tuner* attribute), 23  
`Vocabulary` (class in *matchzoo.preprocessors.units*), 124  
`Vocabulary` (class in *matchzoo.preprocessors.units.vocabulary*), 117  
`Vocabulary.TermIndex` (class in *matchzoo.preprocessors.units*), 124  
`Vocabulary.TermIndex` (class in *matchzoo.preprocessors.units.vocabulary*), 118

## W

`with_word_hashing` (*matchzoo.preprocessors.cdssm\_preprocessor.CDSSMPreprocessor* attribute), 130  
`with_word_hashing` (*matchzoo.preprocessors.CDSSMPreprocessor* attribute), 135  
`with_word_hashing` (*matchzoo.preprocessors.dssm\_preprocessor.DSSMPreprocessor* attribute), 131  
`with_word_hashing` (*matchzoo.preprocessors.DSSMPreprocessor* attribute), 133  
`WordExactMatch` (class in *matchzoo.preprocessors.units*), 125  
`WordExactMatch` (class in *matchzoo.preprocessors.units.word\_exact\_match*), 118  
`WordHashing` (class in *matchzoo.preprocessors.units*), 125  
`WordHashing` (class in *matchzoo.preprocessors.units.word\_hashing*), 119