

---

# MatchZoo Documentation

*Release 1.0*

**MatchZoo**

**Sep 26, 2020**



## 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 . . . . .	11
2.7	KNRM . . . . .	12
2.8	ConvKNRM . . . . .	13
2.9	BiMPM . . . . .	14
2.10	MatchLSTM . . . . .	15
2.11	ArcI . . . . .	16
2.12	ArcII . . . . .	20
2.13	Bert . . . . .	21
2.14	MVLSTM . . . . .	22
2.15	MatchPyramid . . . . .	24
2.16	aNMM . . . . .	25
2.17	HBMP . . . . .	26
2.18	DUET . . . . .	28
2.19	DIIN . . . . .	29
2.20	MatchSRNN . . . . .	32
<b>3</b>	<b>API Reference</b>	<b>33</b>
3.1	matchzoo . . . . .	33
<b>4</b>	<b>Indices and tables</b>	<b>227</b>
	<b>Python Module Index</b>	<b>229</b>
	<b>Index</b>	<b>231</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.



**MATCHZOO**





## MATCHZOO MODEL REFERENCE

### 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	out_activation_func	Activation function used in output layer.		
3	with_embeddings	A flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at <code>padding_idx</code> (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	with_multi_layer_perceptron	A flag to determine whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	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
11	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 5), with a step size of 1
12	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
13	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'
```

(continues on next page)

(continued from previous page)

```
>>> 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 'match-zoo.models.dssm.DSSM'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation_func	Activation function used in output layer.		
3	with_multi_layer_perceptron	A flag to determine whether a multiple layer perceptron is used. Shouldn't be changed.	True	
4	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
5	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
6	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
7	mlp_activation_func	Activation function used in the multiple layer perceptron.	relu	
8	vocab_size	Size of vocabulary.	419	

## 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'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation_func	Activation function used in output layer.		
3	with_multi_layer_perceptron	A flag of whether a multiple layer perceptron is used. Shouldn't be changed.	True	
4	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
5	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
6	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
7	mlp_activation_func	Activation function used in the multiple layer perceptron.	relu	
8	vocab_size	Size of vocabulary.	419	
9	filters	Number of filters in the 1D convolution layer.	3	
10	kernel_size	Number of kernel size in the 1D convolution layer.	3	
11	conv_activation_func	Activation function in the convolution layer.	relu	
12	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	out_activation	Activation function used in output layer.		
3	with_embedding	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	with_multiple_layer_perceptron	Whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	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
11	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
12	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
13	mlp_activation	Activation function used in the multiple layer perceptron.	relu	
14	mask_value	The value to be masked from inputs.	0	
15	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'
```

(continues on next page)

(continued from previous page)

```
>>> 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.drmmtks.DRMMTKS'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_embeddings	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	with_multiple_layer_perceptron	Flag to indicate whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	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
11	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
12	mlp_num_fanout	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
13	mlp_activation	Activation function used in the multiple layer perceptron.	relu	
14	mask_value	The value to be masked from inputs.	0	
15	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	out_activation	Activation function used in output layer.		
3	with_embeddings	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	mask_value	The value to be masked from inputs.	0	
10	dropout	Dropout rate.	0.2	
11	hidden_size	Hidden size.	200	
12	lstm_layer	Number of LSTM layers	1	
13	drop_lstm	Whether dropout LSTM.	False	
14	concat_lstm	Whether concat intermediate outputs.	True	
15	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	out_activation	Activation function used in output layer.		
3	with_embedding_tag	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	kernel_num	The number of RBF kernels.	11	quantitative uniform distribution in [5, 20), with a step size of 1
10	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
11	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 'matchzoo.models.conv_knrm.ConvKNRM'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_embedding	Using help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	filters	The filter size in the convolution layer.	128	
10	conv_activation	The activation function in the convolution layer.	relu	
11	max_ngram	The maximum length of n-grams for the convolution layer.	3	
12	use_crossmatch	Whether to match left n-grams and right n-grams of different lengths	True	
13	kernel_num	The number of RBF kernels.	11	quantitative uniform distribution in [5, 20), with a step size of 1
14	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
15	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
```

(continues on next page)

(continued from previous page)

```
>>> 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.BiMPM'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_embedding	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	mask_value	The value to be masked from inputs.	0	
10	dropout	Dropout rate.	0.2	
11	hidden_size	Hidden size.	100	quantitative uniform distribution in [100, 300), with a step size of 100
12	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	out_activation	Activation function used in output layer.		
3	with_embedding	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	mask_value	The value to be masked from inputs.	0	
10	dropout	Dropout rate.	0.2	
11	hidden_size	Hidden size.	200	
12	lstm_layer	Number of LSTM layers	1	
13	drop_lstm	Whether dropout LSTM.	False	
14	concat_lstm	Whether concat intermediate outputs.	True	
15	rnn_type	Choose rnn type, lstm or gru.	lstm	

## 2.11 ArcI

### 2.11.1 Model Documentation

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'
```

(continues on next page)

(continued from previous page)

```
>>> model.params['dropout_rate'] = 0.5
>>> 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.arci.ArcI'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation_fn	Activation function used in output layer.		
3	with_embedding_help	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	with_multi_layer_perceptron	Flag to determine whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	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
11	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
12	mlp_num_fanout	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
13	mlp_activation_fn	Activation function used in the multiple layer perceptron.	relu	
14	left_length	Length of left input.	10	
15	right_length	Length of right input.	100	
16	conv_activation_fn	The activation function in the convolution layer.	relu	
17	left_filters	The filter size of each convolution blocks for the left input.	[32]	
18	left_kernel_size	The kernel size of each convolution blocks for the left input.	[3]	
19	left_pool_size	The pooling size of each convolution blocks for the left input.	[2]	
20	right_filters	The filter size of each convolution blocks for the right input.	[32]	
21	right_kernel_size	The kernel size of each convolution blocks for the right input.	[3]	
22	right_pool_size	The pooling size of each convolution blocks for the right input.	[2]	
23	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

## 2.12 ArcII

### 2.12.1 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()
```



## 2.12.2 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	out_activation	Activation function used in output layer.		
3	with_embedding	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	left_length	Length of left input.	10	
10	right_length	Length of right input.	100	
11	kernel_1d_count	Kernel count of 1D convolution layer.	32	
12	kernel_1d_size	Kernel size of 1D convolution layer.	3	
13	kernel_2d_count	Kernel count of 2D convolution layer in each block	[32]	
14	kernel_2d_size	Kernel size of 2D convolution layer in each block.	[(3, 3)]	
15	activation	Activation function.	relu	
16	pool_2d_size	Size of pooling layer in each block.	[(2, 2)]	
17	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

## 2.13 Bert

### 2.13.1 Model Documentation

Bert Model.

## 2.13.2 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.bert.Bert'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	mode	Pretrained Bert model.	bert-base-uncased	
4	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

## 2.14 MVLSTM

### 2.14.1 Model Documentation

MVLSTM Model.

#### Examples:

```
>>> model = MVLSTM()
>>> model.params['hidden_size'] = 32
>>> model.params['top_k'] = 50
>>> model.params['mlp_num_layers'] = 2
>>> model.params['mlp_num_units'] = 20
>>> model.params['mlp_num_fan_out'] = 10
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.params['dropout_rate'] = 0.0
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.14.2 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.mvlstm.MVLSTM'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_embeddings	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at <code>padding_idx</code> (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	with_multiple_layer_perceptron	Flag of whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	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
11	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
12	mlp_num_fanout	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
13	mlp_activation	Activation function used in the multiple layer perceptron.	relu	
14	hidden_size	Integer, the hidden size in the bi-directional LSTM layer.	32	
15	num_layers	Integer, number of recurrent layers.	1	
16	top_k	Size of top-k pooling layer.	10	quantitative uniform distribution in [2, 100), with a step size of 1
17	dropout_rate	Float, the dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

## 2.15 MatchPyramid

### 2.15.1 Model Documentation

MatchPyramid Model.

#### Examples:

```
>>> model = MatchPyramid()
>>> model.params['embedding_output_dim'] = 300
>>> model.params['kernel_count'] = [16, 32]
>>> model.params['kernel_size'] = [[3, 3], [3, 3]]
>>> model.params['dpool_size'] = [3, 10]
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

### 2.15.2 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.match_pyramid.MatchPyramid'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_embedding_help	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	kernel_count	The kernel count of the 2D convolution of each block.	[32]	
10	kernel_size	The kernel size of the 2D convolution of each block.	[[3, 3]]	
11	activation	The activation function.	relu	
12	dpool_size	The max-pooling size of each block.	[3, 10]	
13	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

## 2.16 aNMM

### 2.16.1 Model Documentation

aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model.

#### Examples:

```
>>> model = aNMM()
>>> model.params['embedding_output_dim'] = 300
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

### 2.16.2 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.anmm.aNMM'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_embedding_help	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at <code>padding_idx</code> (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	mask_value	The value to be masked from inputs.	0	
10	num_bins	Integer, number of bins.	200	
11	hidden_sizes	Number of hidden size for each hidden layer	[100]	
12	activation	The activation function.	relu	
13	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

## 2.17 HBMP

### 2.17.1 Model Documentation

HBMP model.

**Examples:**

```
>>> model = HBMP ()
>>> model.params['embedding_input_dim'] = 200
>>> model.params['embedding_output_dim'] = 100
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 10
>>> model.params['mlp_num_fan_out'] = 10
>>> model.params['mlp_activation_func'] = nn.LeakyReLU(0.1)
>>> model.params['lstm_hidden_size'] = 5
>>> model.params['lstm_num'] = 3
>>> model.params['num_layers'] = 3
>>> model.params['dropout_rate'] = 0.1
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.17.2 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.hbmp.HBMP'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_embeddings	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector atpadding_idx (initialized to zeros) whenever it encountersthe index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	with_multiple_layer_perceptron	Flag to indicate whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	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
11	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
12	mlp_num_fanout	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
13	mlp_activation	Activation function used in the multiple layer perceptron.	relu	
14	lstm_hidden_size	Integer, the hidden size of the bi-directional LSTM layer.	5	
15	lstm_num	Integer, number of LSTM units	3	
16	num_layers	Integer, number of LSTM layers.	1	
17	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

## 2.18 DUET

### 2.18.1 Model Documentation

Duet Model.

**Examples:**

```
>>> model = DUET()
>>> model.params['left_length'] = 10
>>> model.params['right_length'] = 40
>>> model.params['lm_filters'] = 300
>>> model.params['mlp_num_layers'] = 2
>>> model.params['mlp_num_units'] = 300
>>> model.params['mlp_num_fan_out'] = 300
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.params['vocab_size'] = 2000
>>> model.params['dm_filters'] = 300
>>> model.params['dm_conv_activation_func'] = 'relu'
>>> model.params['dm_kernel_size'] = 3
>>> model.params['dm_right_pool_size'] = 8
>>> model.params['dropout_rate'] = 0.5
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```



## 2.18.2 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.duet.DUET'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_multi_layer_perceptron	Whether a multiple layer perceptron is used. Shouldn't be changed.	True	
4	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
5	mlp_num_layers	Number of layers of the multiple layer perceptron.	3	quantitative uniform distribution in [1, 6), with a step size of 1
6	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
7	mlp_activation	Activation function used in the multiple layer perceptron.	relu	
8	mask_value	The value to be masked from inputs.	0	
9	left_length	Length of left input.	10	
10	right_length	Length of right input.	40	
11	lm_filters	Filter size of 1D convolution layer in the local model.	300	
12	vocab_size	Vocabulary size of the tri-letters used in the distributed model.	419	
13	dm_filters	Filter size of 1D convolution layer in the distributed model.	300	
14	dm_kernel_size	Kernel size of 1D convolution layer in the distributed model.	3	
15	dm_conv_activation	Activation functions of the convolution layer in the distributed model.	relu	
16	dm_right_pool_kernel_size	Kernel size of 1D convolution layer in the distributed model.	8	
17	dropout_rate	The dropout rate.	0.5	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.02

## 2.19 DIIN

### 2.19.1 Model Documentation

DIIN model.

**Examples:**

```
>>> model = DIIN()
>>> model.params['embedding_input_dim'] = 10000
>>> model.params['embedding_output_dim'] = 300
>>> model.params['mask_value'] = 0
>>> model.params['char_embedding_input_dim'] = 100
>>> model.params['char_embedding_output_dim'] = 8
>>> model.params['char_conv_filters'] = 100
>>> model.params['char_conv_kernel_size'] = 5
>>> model.params['first_scale_down_ratio'] = 0.3
>>> model.params['nb_dense_blocks'] = 3
>>> model.params['layers_per_dense_block'] = 8
>>> model.params['growth_rate'] = 20
>>> model.params['transition_scale_down_ratio'] = 0.5
>>> model.params['conv_kernel_size'] = (3, 3)
>>> model.params['pool_kernel_size'] = (2, 2)
>>> model.params['dropout_rate'] = 0.2
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

## 2.19.2 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.diin	DIIN'>
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_embedding	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	mask_value	The value to be masked from inputs.	0	
10	char_embedding_input_dim	The input dimension of character embedding layer.	100	
11	char_embedding_output_dim	The output dimension of character embedding layer.	8	
12	char_conv_filter	The filter size of character convolution layer.	100	
13	char_conv_kernel	The kernel size of character convolution layer.	5	
14	first_scale_down	The channel scale down ratio of the convolution layer before densenet.	0.3	
15	nb_dense_blocks	The number of blocks in densenet.	3	
16	layers_per_dense_block	The number of convolution layers in dense block.	8	
17	growth_rate	The filter size of each convolution layer in dense block.	20	
18	transition_scale_down	The channel scale down ratio of the convolution layer in transition block.	0.5	
19	conv_kernel_size	The kernel size of convolution layer in dense block.	(3, 3)	
20	pool_kernel_size	The kernel size of pooling layer in transition block.	(2, 2)	
21	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

## 2.20 MatchSRNN

### 2.20.1 Model Documentation

Match-SRNN Model.

#### Examples:

```
>>> model = MatchSRNN()
>>> model.params['channels'] = 4
>>> model.params['units'] = 10
>>> model.params['dropout'] = 0.2
>>> model.params['direction'] = 'lt'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

### 2.20.2 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.match_srnn.MatchSRNN'>	
1	task	Decides model output shape, loss, and metrics.		
2	out_activation	Activation function used in output layer.		
3	with_embedding	Flag used help <i>auto</i> module. Shouldn't be changed.	True	
4	embedding	FloatTensor containing weights for the Embedding.		
5	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
6	embedding_output_dim	Should be set manually.		
7	padding_idx	If given, pads the output with the embedding vector at padding_idx (initialized to zeros) whenever it encounters the index.	0	
8	embedding_freeze	<i>True</i> to freeze embedding layer training, <i>False</i> to enable embedding parameters.	False	
9	channels	Number of word interaction tensor channels	4	
10	units	Number of SpatialGRU units	10	
11	direction	Direction of SpatialGRU scanning	lt	
12	dropout	The dropout rate.	0.2	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

## API REFERENCE

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

##### Functions

---

<code>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)</code>	A simple shorthand for using <code>matchzoo.Preparer</code> .
--	---

---

`matchzoo.auto.preparer.prepare.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`.

---

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

*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.preparer.preparer`

## Module Contents

### Classes

---

*Preparer*

Unified setup processes of all MatchZoo models.

---

**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
>>> train_raw = mz.datasets.toy.load_data('train', 'ranking')
>>> model, prpr, dsb, dlb = preparer.prepare(model_class,
...                                       train_raw)
>>> model.params.completed(exclude=['out_activation_func'])
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) → typing.Tuple[BaseModel, BasePreprocessor, DatasetBuilder, DataLoaderBuilder]  
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*) → typing.Tuple[BaseModel, np.ndarray]

**\_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*) → dict  
Default config getter.

## Package Contents

### Classes

---

*Preparer*

---

Unified setup processes of all MatchZoo models.

---

## Functions

---

*prepare*

---

**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(exclude=['out_activation_func'])
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`) → `typing.Tuple[BaseModel, BasePreprocessor, DatasetBuilder, DataLoaderBuilder]`

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*) → typing.Tuple[BaseModel, np.ndarray]

**`_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*) → dict

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

## Functions

---

```
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)
```

---

```
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

### Classes

---

*Tuner*

Model hyper-parameters tuner.

---

**class** `matchzoo.auto.tuner.tuner.Tuner` (*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*: `int = 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)

**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*)

**property params** (*self*)

*params* getter.

**property trainloader** (*self*)

*trainloader* getter.

**property validloader** (*self*)

*validloader* getter.

**property fit\_kwargs** (*self*)

*fit\_kwargs* getter.

**property metric** (*self*)

*metric* getter.

**property mode** (*self*)

*mode* getter.

**property num\_runs** (*self*)

*num\_runs* getter.

**property verbose** (*self*)

*verbose* getter.

```

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

### Classes

---

<i>Tuner</i>	Model hyper-parameters tuner.
--------------	-------------------------------

---

### Functions

---

<i>tune</i>
-------------

---

```

class matchzoo.auto.tuner.Tuner (params: mz.ParamTable, optimizer: str = 'adam', train-
loader: mz.data_loader.DataLoader = None, validloader:
mz.data_loader.DataLoader = None, embedding: np.ndarray =
None, fit_kwargs: dict = None, metric: typing.Union[str, Base-
Metric] = 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)

**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*)

**property params** (*self*)

*params* getter.

**property trainloader** (*self*)

*trainloader* getter.

**property validloader** (*self*)

*validloader* getter.

**property fit\_kwargs** (*self*)

*fit\_kwargs* getter.

**property metric** (*self*)

*metric* getter.

**property mode** (*self*)

*mode* getter.

**property num\_runs** (*self*)

*num\_runs* getter.

**property verbose** (*self*)

*verbose* getter.

**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, 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'
```

(continues on next page)

(continued from previous page)

```

>>> 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']

```

## Package Contents

### Classes

<i>Preparer</i>	Unified setup processes of all MatchZoo models.
<i>Tuner</i>	Model hyper-parameters tuner.

**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(exclude=['out_activation_func'])
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) → typing.Tuple[BaseModel, BasePreprocessor, DatasetBuilder, DataLoaderBuilder]  
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*) → typing.Tuple[BaseModel, np.ndarray]

**\_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*) → dict  
Default config getter.

```
class matchzoo.auto.Tuner (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)
```

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)

**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*)

**property params** (*self*)

*params* getter.

**property trainloader** (*self*)

*trainloader* getter.

**property validloader** (*self*)

*validloader* getter.

**property fit\_kwargs** (*self*)

*fit\_kwargs* getter.

**property metric** (*self*)

*metric* getter.

**property mode** (*self*)

*mode* getter.

**property num\_runs** (*self*)

*num\_runs* getter.

**property verbose** (*self*)

*verbose* getter.

**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)

```

matchzoo.data\_pack

## Submodules

matchzoo.data\_pack.data\_pack

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

## Module Contents

### Classes

---

<i>DataPack</i>	Matchzoo <i>DataPack</i> data structure, store dataframe and context.
-----------------	---

---

### Functions

---

<i>_convert_to_list_index</i> (index: typing.Union[int, slice, np.array], length: int)	typ-
<i>load_data_pack</i> (dirpath: typing.Union[str, Path]) → DataPack	Load a <i>DataPack</i> . The reverse function of <i>save()</i> .

---

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]`) → `pd.DataFrame`  
Slicer.

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

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

**property** `has_label` (*self*) → `bool`

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

`__len__` (*self*) → `int`  
Get number of rows in the class:`DataPack` object.

**property** `frame` (*self*) → `'DataPack.FrameView'`

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

```

**unpack** (*self*) → typing.Tuple[typing.Dict[str, np.array], typing.Optional[np.array]]  
 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]) → 'DataPack'  
 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*.

**property** *relation* (*self*)  
*relation* getter.

**property** *left* (*self*) → pd.DataFrame  
 Get *left()* of *DataPack*.

**property** *right* (*self*) → pd.DataFrame  
 Get *right()* of *DataPack*.

**copy** (*self*) → 'DataPack'

**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.

**drop\_empty** (*self*)

Process empty data by removing corresponding rows.

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

**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]*) → *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`

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

## Module Contents

### Functions

---

`pack`(*df*: *pd.DataFrame*, *task*: *typing.Union[str, BaseTask]* = 'ranking') → 'matchzoo.DataPack'

---

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

---

`_gen_ids`(*data*: *pd.DataFrame*, *col*: *str*, *prefix*: *str*)

---

`matchzoo.data_pack.pack.pack` (*df*: *pd.DataFrame*, *task*: *typing.Union[str, BaseTask]* = 'ranking') → '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.
- **task** – Could be one of `ranking`, `classification` or a `matchzoo.engine.BaseTask` instance.



**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, task='classification').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
>>> mz.pack(df, task='ranking').frame()
  id_left text_left id_right text_right  label
0     L-0         A      R-0         a    0.0
1     L-0         A      R-1         b    1.0
2     L-1         B      R-1         b    1.0
3     L-2         C      R-2         c    0.0

```

`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****Classes**

---

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

---

**Functions**

---

*load\_data\_pack*(*dirpath*: `typing.Union[str, Path]`) Load a *DataPack*. The reverse function of `save()`.  
→ *DataPack*

---

*pack*(*df*: `pd.DataFrame`, *task*: `typing.Union[str, Base-Task]` = 'ranking') → 'matchzoo.DataPack'

---

**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]`) → `pd.DataFrame`  
Slicer.

**\_\_call\_\_** (*self*)  
**Returns** A full copy. Equivalent to `frame[:]`.

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

**property** `has_label` (*self*) → `bool`

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

**\_\_len\_\_** (*self*) → `int`  
Get numer of rows in the class:`DataPack` object.

**property** `frame` (*self*) → `'DataPack.FrameView'`  
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

```

**unpack** (*self*) → typing.Tuple[typing.Dict[str, np.array], typing.Optional[np.array]]  
 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]) → 'DataPack'  
 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*.

**property** *relation* (*self*)  
*relation* getter.

**property** *left* (*self*) → pd.DataFrame  
 Get *left()* of *DataPack*.

**property** *right* (*self*) → pd.DataFrame  
 Get *right()* of *DataPack*.

**copy** (*self*) → 'DataPack'

**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.

**drop\_empty** (*self*)

Process empty data by removing corresponding rows.

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

**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*, *task*: *typing.Union[str, BaseTask]* = 'ranking') → *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.
- **task** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance.

**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, task='classification').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
>>> mz.pack(df, task='ranking').frame()
  id_left text_left id_right text_right  label
0     L-0         A       R-0         a    0.0
1     L-0         A       R-1         b    1.0
2     L-1         B       R-1         b    1.0
3     L-2         C       R-2         c    0.0
```

`matchzoo.dataloader`

## Subpackages

`matchzoo.dataloader.callbacks`

## Submodules

`matchzoo.dataloader.callbacks.histogram`

## Module Contents

### Classes

---

<i>Histogram</i>	Generate data with matching histogram.
------------------	--

---

### Functions

---

<code>_trunc_text(input_text: list, length: list) → list</code>	Truncating the input text according to the input length.
<code>_build_match_histogram(x: dict, match_hist_unit: mz.preprocessors.units.MatchingHistogram) → np.ndarray</code>	Generate the matching histogram for input.

---

**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.MatchingHistogramUnit)
                                                             → np.ndarray
```

Generate the matching histogram 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

### Classes

---

<code>LambdaCallback</code>	<code>LambdaCallback</code> . Just a shorthand for creating a callback class.
-----------------------------	---

---

```
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.ngram`

## Module Contents

### Classes

---

<i>Ngram</i>	Generate the character n-gram for data.
--------------	---

---

### Functions

---

<code>_build_word_ngram_map</code> ( <code>ngram_process_unit</code> : <code>mz.preprocessors.units.NgramLetter</code> , <code>ngram_vocab_unit</code> : <code>mz.preprocessors.units.Vocabulary</code> , <code>index_term</code> : dict, <code>mode</code> : str = 'index') → dict	Generate the word to ngram vector mapping.
---	--

---

**class** `matchzoo.dataloader.callbacks.ngram.Ngram` (*preprocessor*:  
*mz.preprocessors.BasicPreprocessor*,  
*mode*: str = 'index')

Bases: `matchzoo.engine.base_callback.BaseCallback`

Generate the character n-gram for data.

#### Parameters

- **preprocessor** – The fitted `BasePreprocessor` object, which contains the n-gram units information.
- **mode** – It can be one of 'index', 'onehot', 'sum' or 'aggregate'.

#### Example

```
>>> import matchzoo as mz
>>> from matchzoo.dataloader.callbacks import Ngram
>>> data = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.BasicPreprocessor(ngram_size=3)
>>> data = preprocessor.fit_transform(data)
>>> callback = Ngram(preprocessor=preprocessor, mode='index')
>>> dataset = mz.dataloader.Dataset(
...     data, callbacks=[callback])
>>> _ = dataset[0]
```

**on\_batch\_unpacked** (*self*, *x*, *y*)  
Insert `ngram_left` and `ngram_right` to *x*.

`matchzoo.dataloader.callbacks.ngram._build_word_ngram_map` (*ngram\_process\_unit*:  
*mz.preprocessors.units.NgramLetter*,  
*ngram\_vocab\_unit*:  
*mz.preprocessors.units.Vocabulary*,  
*index\_term*: dict, *mode*:  
str = 'index') → dict

Generate the word to ngram vector mapping.

#### Parameters

- **ngram\_process\_unit** – The fitted NgramLetter object.
- **ngram\_vocab\_unit** – The fitted Vocabulary object.
- **index\_term** – The index to term mapping dict.
- **mode** – It be one of ‘index’, ‘onehot’, ‘sum’ or ‘aggregate’.

**Returns** the word to ngram vector mapping.

`matchzoo.dataloader.callbacks.padding`

## Module Contents

### Classes

<i>BasicPadding</i>	Pad data for basic preprocessor.
<i>DRMMPadding</i>	Pad data for DRMM Model.
<i>BertPadding</i>	Pad data for bert preprocessor.

### Functions

<i>_infer_dtype</i> (value)	Infer the dtype for the features.
<i>_padding_2D</i> (input, output, mode: str = ‘pre’)	Pad the input 2D-tensor to the output 2D-tensor.
<i>_padding_3D</i> (input, output, mode: str = ‘pre’)	Pad the input 3D-tensor to the output 3D-tensor.

`matchzoo.dataloader.callbacks.padding._infer_dtype` (value)

Infer the dtype for the features.

It is required as the input is usually array of objects before padding.

`matchzoo.dataloader.callbacks.padding._padding_2D` (input, output, mode: str = ‘pre’)

Pad the input 2D-tensor to the output 2D-tensor.

#### Parameters

- **input** – The input 2D-tensor contains the origin values.
- **output** – The output is a shapped 2D-tensor which have filled with pad value.
- **mode** – The padding model, which can be ‘pre’ or ‘post’.

`matchzoo.dataloader.callbacks.padding._padding_3D` (input, output, mode: str = ‘pre’)

Pad the input 3D-tensor to the output 3D-tensor.

#### Parameters

- **input** – The input 3D-tensor contains the origin values.
- **output** – The output is a shapped 3D-tensor which have filled with pad value.
- **mode** – The padding model, which can be ‘pre’ or ‘post’.

```
class matchzoo.dataloader.callbacks.padding.BasicPadding (fixed_length_left: int = None, fixed_length_right: int = None, pad_word_value: typing.Union[int, str] = 0, pad_word_mode: str = 'pre', with_ngram: bool = False, fixed_ngram_length: int = None, pad_ngram_value: typing.Union[int, str] = 0, pad_ngram_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\_word\_value** – the value to fill text.
- **pad\_word\_mode** – String, `pre` or `post`: pad either before or after each sequence.
- **with\_ngram** – Boolean. Whether to pad the n-grams.
- **fixed\_ngram\_length** – Integer. If set, each word will be padded to this length, or it will be set as the maximum length of words in current batch.
- **pad\_ngram\_value** – the value to fill empty n-grams.
- **pad\_ngram\_mode** – String, `pre` or `post`: pad either before of 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.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

### Classes

<i>LambdaCallback</i>	LambdaCallback. Just a shorthand for creating a callback class.
<i>Histogram</i>	Generate data with matching histogram.
<i>Ngram</i>	Generate the character n-gram for data.
<i>BasicPadding</i>	Pad data for basic preprocessor.
<i>DRMMPadding</i>	Pad data for DRMM Model.
<i>BertPadding</i>	Pad data for bert preprocessor.

```
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.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.Ngram` (*preprocessor*: *mz.preprocessors.BasicPreprocessor*,  
*mode*: *str* = 'index')

Bases: `matchzoo.engine.base_callback.BaseCallback`

Generate the character n-gram for data.

#### Parameters

- **preprocessor** – The fitted BasePreprocessor object, which contains the n-gram units information.
- **mode** – It can be one of 'index', 'onehot', 'sum' or 'aggregate'.

### Example

```
>>> import matchzoo as mz
>>> from matchzoo.dataloader.callbacks import Ngram
>>> data = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.BasicPreprocessor(ngram_size=3)
>>> data = preprocessor.fit_transform(data)
>>> callback = Ngram(preprocessor=preprocessor, mode='index')
>>> dataset = mz.dataloader.Dataset(
...     data, callbacks=[callback])
>>> _ = dataset[0]
```

`on_batch_unpacked` (*self*, *x*, *y*)  
 Insert *ngram\_left* and *ngram\_right* to *x*.

**class** `matchzoo.dataloader.callbacks.BasicPadding` (*fixed\_length\_left*: *int* = None,  
*fixed\_length\_right*: *int* = None,  
*pad\_word\_value*: *typing.Union[int, str]* = 0, *pad\_word\_mode*: *str* =  
'pre', *with\_ngram*: *bool* = False,  
*fixed\_ngram\_length*: *int* = None,  
*pad\_ngram\_value*: *typing.Union[int, str]* = 0, *pad\_ngram\_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\_word\_value** – the value to fill text.
- **pad\_word\_mode** – String, *pre* or *post*: pad either before or after each sequence.
- **with\_ngram** – Boolean. Whether to pad the n-grams.
- **fixed\_ngram\_length** – Integer. If set, each word will be padded to this length, or it will be set as the maximum length of words in current batch.
- **pad\_ngram\_value** – the value to fill empty n-grams.
- **pad\_ngram\_mode** – String, *pre* or *post*: pad either before of 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.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

### Classes

---

*DataLoader*

DataLoader that loads batches of data from a Dataset.

---

```
class matchzoo.dataloader.dataloader.DataLoader(dataset: Dataset, device: typing.Union[torch.device, int, list, None] = None, stage='train', 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.
- **device** – The desired device of returned tensor. Default: if None, use the current device. If `torch.device` or int, use device specified by user. If list, the first item will be used.
- **stage** – One of “train”, “dev”, and “test”. (default: “train”)
- **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.BasicPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset = mz.dataloader.Dataset(
...     data_processed, mode='point', batch_size=32)
>>> padding_callback = mz.dataloader.callbacks.BasicPadding()
>>> dataloader = mz.dataloader.DataLoader(
```

(continues on next page)

(continued from previous page)

```
...     dataset, stage='train', callback=padding_callback)
>>> len(dataloader)
4
```

`__len__(self) → int`

Get the total number of batches.

**property** `id_left(self) → np.ndarray`

`id_left` getter.

**property** `label(self) → np.ndarray`

`label` getter.

`__iter__(self) → typing.Tuple[dict, torch.tensor]`

Iteration.

`__handle_callbacks_on_batch_unpacked(self, x, y)`

`matchzoo.dataloader.dataloader_builder`

## Module Contents

### Classes

---

`DataLoaderBuilder`

DataLoader Bulider. In essence a wrapped partial function.

---

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

Bases: `object`

DataLoader Bulider. In essence a wrapped partial function.

### Example

```
>>> import matchzoo as mz
>>> padding_callback = mz.dataloader.callbacks.BasicPadding()
>>> builder = mz.dataloader.DataLoaderBuilder(
...     stage='train', callback=padding_callback
... )
>>> data_pack = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.BasicPreprocessor()
>>> 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`) → `DataLoader`

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

### Classes

---

*Dataset*

Dataset that is built from a data pack.

---

```
class matchzoo.dataloader.dataset.Dataset (data_pack: mz.DataPack, mode='point',
num_dup: int = 1, num_neg: int = 1,
batch_size: int = 32, resample: bool =
False, shuffle: bool = True, sort: bool = False,
callbacks: typing.List[BaseCallback] = None)
```

Bases: torch.utils.data.IterableDataset

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)
- **batch\_size** – Batch size. (default: 32)
- **resample** – Either to resample for each epoch, only effective when *mode* is “pair”. (default: *True*)
- **shuffle** – Either to shuffle the samples/instances. (default: *True*)
- **sort** – Whether to sort data according to length\_right. (default: *False*)
- **callbacks** – Callbacks. See *matchzoo.dataloader.callbacks* for more details.

## Examples

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data(stage='train')
>>> preprocessor = mz.preprocessors.BasicPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset_point = mz.dataloader.Dataset(
...     data_processed, mode='point', batch_size=32)
>>> len(dataset_point)
```

4

(continues on next page)

(continued from previous page)

```

>>> dataset_pair = mz.dataloader.Dataset(
...     data_processed, mode='pair', num_dup=2, num_neg=2, batch_size=32)
>>> len(dataset_pair)
1

```

**\_\_getitem\_\_** (*self*, *item*) → typing.Tuple[dict, np.ndarray]  
Get a batch from index *idx*.

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

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

**\_\_iter\_\_** (*self*)  
Create a generator that iterate over the Batches.

**on\_epoch\_end** (*self*)  
Reorganize the index array if needed.

**resample\_data** (*self*)  
Reorganize data.

**reset\_index** (*self*)  
Set the `_batch_indices`.

Here the `_batch_indices` records the index of all the instances.

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

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

**property callbacks** (*self*)  
*callbacks* getter.

**property num\_neg** (*self*)  
*num\_neg* getter.

**property num\_dup** (*self*)  
*num\_dup* getter.

**property mode** (*self*)  
*mode* getter.

**property batch\_size** (*self*)  
*batch\_size* getter.

**property shuffle** (*self*)  
*shuffle* getter.

**property sort** (*self*)  
*sort* getter.

**property resample** (*self*)  
*resample* getter.

**property batch\_indices** (*self*)  
*batch\_indices* getter.

**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

### Classes

---

<i>DatasetBuilder</i>	Dataset Bulider. In essence a wrapped partial function.
-----------------------	---

---

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

Bases: `object`

Dataset Bulider. In essence 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*) → `Dataset`

Build a Dataset.

#### Parameters

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

## Package Contents

### Classes

---

<i>Dataset</i>	Dataset that is built from a data pack.
<i>DataLoader</i>	<code>DataLoader</code> that loads batches of data from a <code>Dataset</code> .
<i>DataLoaderBuilder</i>	<code>DataLoader</code> Bulider. In essence a wrapped partial function.
<i>DatasetBuilder</i>	Dataset Bulider. In essence a wrapped partial function.

---

**class** `matchzoo.dataloader.Dataset` (*data\_pack: mz.DataPack, mode='point', num\_dup: int = 1, num\_neg: int = 1, batch\_size: int = 32, resample: bool = False, shuffle: bool = True, sort: bool = False, callbacks: typing.List[BaseCallback] = None*)

Bases: `torch.utils.data.IterableDataset`

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)
- **batch\_size** – Batch size. (default: 32)
- **resample** – Either to resample for each epoch, only effective when *mode* is “pair”. (default: *True*)
- **shuffle** – Either to shuffle the samples/instances. (default: *True*)
- **sort** – Whether to sort data according to *length\_right*. (default: *False*)
- **callbacks** – Callbacks. See *matchzoo.dataloader.callbacks* for more details.

## Examples

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data(stage='train')
>>> preprocessor = mz.preprocessors.BasicPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset_point = mz.dataloader.Dataset(
...     data_processed, mode='point', batch_size=32)
>>> len(dataset_point)
4
>>> dataset_pair = mz.dataloader.Dataset(
...     data_processed, mode='pair', num_dup=2, num_neg=2, batch_size=32)
>>> len(dataset_pair)
1
```

`__getitem__` (*self*, *item*) → typing.Tuple[dict, np.ndarray]  
Get a batch from index *idx*.

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

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

`__iter__` (*self*)  
Create a generator that iterate over the Batches.

`on_epoch_end` (*self*)  
Reorganize the index array if needed.

`resample_data` (*self*)  
Reorganize data.

`reset_index` (*self*)  
Set the `_batch_indices`.

Here the `_batch_indices` records the index of all the instances.

`_handle_callbacks_on_batch_data_pack` (*self*, *batch\_data\_pack*)

`_handle_callbacks_on_batch_unpacked` (*self*, *x*, *y*)

**property callbacks** (*self*)  
*callbacks* getter.

**property num\_neg** (*self*)  
*num\_neg* getter.

**property num\_dup** (*self*)  
*num\_dup* getter.

**property mode** (*self*)  
*mode* getter.

**property batch\_size** (*self*)  
*batch\_size* getter.

**property shuffle** (*self*)  
*shuffle* getter.

**property sort** (*self*)  
*sort* getter.

**property resample** (*self*)  
*resample* getter.

**property batch\_indices** (*self*)  
*batch\_indices* getter.

**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.data_loader.DataLoader` (*dataset*: *Dataset*, *device*: *typing.Union[torch.device, int, list, None]* = *None*, *stage*='train', *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.
- **device** – The desired device of returned tensor. Default: if `None`, use the current device. If `torch.device` or `int`, use device specified by user. If list, the first item will be used.
- **stage** – One of “train”, “dev”, and “test”. (default: “train”)
- **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.BasicPreprocessor()
>>> data_processed = preprocessor.fit_transform(data_pack)
>>> dataset = mz.dataloader.Dataset(
...     data_processed, mode='point', batch_size=32)
>>> padding_callback = mz.dataloader.callbacks.BasicPadding()
>>> dataloader = mz.dataloader.DataLoader(
...     dataset, stage='train', callback=padding_callback)
>>> len(dataloader)
4
```

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

**property** `id_left` (*self*) → np.ndarray  
`id_left` getter.

**property** `label` (*self*) → np.ndarray  
`label` getter.

`__iter__` (*self*) → typing.Tuple[dict, torch.tensor]  
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.BasicPadding()
>>> builder = mz.dataloader.DataLoaderBuilder(
...     stage='train', callback=padding_callback
... )
>>> data_pack = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.BasicPreprocessor()
>>> 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) → *DataLoader*  
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 Builder. In essence 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*) → *Dataset*  
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

#### Functions

---

`load_fasttext_embedding`(*language*: *str* = 'en') = Return the pretrained fasttext embedding.  
→ `mz.embedding.Embedding`

---

`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.Emb`

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

### Functions

---

`load_glove_embedding`(dimension: int = 50) → Return the pretrained glove embedding.  
`mz.embedding.Embedding`

---

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

`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

### Functions

---

`load_glove_embedding`(dimension: int = 50) → Return the pretrained glove embedding.  
`mz.embedding.Embedding`

---

`load_fasttext_embedding`(language: str = 'en') → Return the pretrained fasttext embedding.  
`mz.embedding.Embedding`

---

`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

### Functions

---

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

---

```
_download_data()
```

---

```
_read_data(path, stage, task)
```

---

`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: typing.Union[str, BaseTask] = '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, task)`

## Package Contents

### Functions

---

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

---

```
matchzoo.datasets.quora_qp.load_data(stage: str = 'train', task: typing.Union[str, BaseTask] = '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

#### Functions

---

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

---

```
_download_data()
```

---

```
_read_data(path, task, target_label)
```

---

```
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: typing.Union[str, BaseTask] = 'classification', target_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: *classification*)
- **target\_label** – If *ranking*, chose one of *entailment*, *contradiction* and *neutral* 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, task, target_label)
```

## Package Contents

### Functions

---

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

---

```
matchzoo.datasets.snli.load_data(stage: str = 'train', task: typing.Union[str, BaseTask] = 'classification', target_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: *classification*)
- **target\_label** – If *ranking*, chose one of *entailment*, *contradiction* and *neutral* 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

### Classes

---

<i>BaseTask</i>	Base Task, shouldn't be used directly.
-----------------	--

---

### Functions

---

*load\_data*(stage: str = 'train', task: typing.Union[str, BaseTask] = 'ranking', return\_classes: bool = False) → typing.Union[matchzoo.DataPack, typing.Tuple[matchzoo.DataPack, list]]

---

*load\_embedding*()

---

**class** matchzoo.datasets.toy.**BaseTask** (*losses=None, metrics=None*)

Bases: abc.ABC

Base Task, shouldn't be used directly.

**TYPE = base**

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

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

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

**property losses** (*self*)

**Returns** Losses used in the task.

**property metrics** (*self*)

**Returns** Metrics used in the task.

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

**Returns** a list of available losses.

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

**Returns** a list of available metrics.

**property output\_shape** (*self*) → tuple

**Returns** output shape of a single sample of the task.

**property output\_dtype** (*self*)

**Returns** output data type for specific task.

matchzoo.datasets.toy.**load\_data** (*stage: str = 'train', task: typing.Union[str, BaseTask] = 'ranking', return\_classes: bool = False*) → typing.Union[matchzoo.DataPack, typing.Tuple[matchzoo.DataPack, list]]

Load toy 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

#### Functions

---

`load_data`(stage: str = 'train', task: typing.Union[str, BaseTask] = 'ranking', filtered: bool = False, return\_classes: bool = False) → typing.Union[matchzoo.DataPack, tuple]

---

`_download_data()`

---

`_read_data`(path, task)

---

`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: typing.Union[str, BaseTask] = 'ranking', filtered: 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.

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

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

## Package Contents

### Functions

---

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

---

```
matchzoo.datasets.wiki_qa.load_data(stage: str = 'train', task: typing.Union[str, BaseTask] = 'ranking', filtered: 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

### Functions

---

```
list_available()
```

---

```
matchzoo.datasets.list_available()
```

`matchzoo.embedding`

## Submodules

`matchzoo.embedding.embedding`

Matchzoo toolkit for token embedding.

## Module Contents

### Classes

---

<code>Embedding</code>	Embedding class.
------------------------	------------------

---

### Functions

---

<code>load_from_file(file_path: str, mode: str = 'word2vec')</code>	Load embedding from <i>file_path</i> . → <code>Embedding</code>
---	--

---

**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]*)→ `np.ndarray`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****Classes**

---

*Embedding*

---

Embedding class.

---

**Functions**

---

*load\_from\_file*(*file\_path*: *str*, *mode*: *str* = 'word2vec') → *Embedding*

---

**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]*)  
 → *np.ndarray*  
 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

### Classes

---

*BaseCallback*

DataGenerator callback base class.

---

**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.

**abstract 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

### Classes

---

<code>BaseMetric</code>	Metric base class.
<code>RankingMetric</code>	Ranking metric base class.
<code>ClassificationMetric</code>	Ranking metric base class.

---

### Functions

---

<code>sort_and_couple</code> ( <i>labels</i> : <i>np.array</i> , <i>scores</i> : <i>Zip the labels with scores into a single list.</i> ) <i>np.array</i> → <i>np.array</i>
---

---

**class** `matchzoo.engine.base_metric.BaseMetric`

Bases: `abc.ABC`

Metric base class.

**ALIAS** = `base_metric`

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

**abstract** `__repr__`(*self*)

**Returns** Formated 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

### Classes

---

`BaseModel`

Abstract base class of all MatchZoo models.

---

**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()
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
```

**classmethod** `get_default_params` (*cls*, *with\_embedding=False*, *with\_multi\_layer\_perceptron=False*) → ParamTable

Model default parameters.

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

## 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*, *truncated\_mode*: str = 'pre', *truncated\_length\_left*: typing.Optional[int] = None, *truncated\_length\_right*: typing.Optional[int] = None, *filter\_mode*: str = 'df', *filter\_low\_freq*: float = 1, *filter\_high\_freq*: float = float('inf'), *remove\_stop\_words*: bool = False, *ngram\_size*: typing.Optional[int] = None) → BasePreprocessor

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*, *fixed\_length\_left*: int = None, *fixed\_length\_right*: int = None, *pad\_word\_value*: typing.Union[int, str] = 0, *pad\_word\_mode*: str = 'pre', *with\_ngram*: bool = False, *fixed\_ngram\_length*: int = None, *pad\_ngram\_value*: typing.Union[int, str] = 0, *pad\_ngram\_mode*: str = 'pre') → BaseCallback

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.

**`property params`** (*self*) → ParamTable

**Returns** model parameters.

**`abstract build`** (*self*)

Build model, each subclass need to implement this method.

**`abstract 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*) → nn.Module

**Returns** an embedding module.

**`_make_default_embedding_layer`** (*self*, *\*\*kwargs*) → nn.Module

**Returns** an embedding module.

**`_make_output_layer`** (*self*, *in\_features*: int = 0) → nn.Module

**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()) → nn.Module

**Returns** a perceptron layer.

**`_make_multi_layer_perceptron_layer`** (*self*, *in\_features*) → nn.Module

**Returns** a multiple layer perceptron.

## matchzoo.engine.base\_preprocessor

*BasePreprocessor* define input and ouput for processors.

## Module Contents

### Classes

---

<i>BasePreprocessor</i>	<i>BasePreprocessor</i> to input handle data.
-------------------------	---

---

### Functions

---

<i>validate_context</i> (func)	Validate context in the preprocessor.
<i>load_preprocessor</i> (dirpath: typing.Union[str, Path]) → 'mz.DataPack'	Load the fitted <i>context</i> . The reverse function of <i>save</i> ().

---

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**

**property context** (*self*)  
 Return context.

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

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

### Classes

---

*BaseTask*

Base Task, shouldn't be used directly.

---

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

Bases: `abc.ABC`

Base Task, shouldn't be used directly.

**TYPE = base**

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

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

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

**property losses** (*self*)

**Returns** Losses used in the task.

**property metrics** (*self*)

**Returns** Metrics used in the task.

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

**Returns** a list of available losses.

**abstract classmethod** `list_available_metrics` (*cls*) → list

**Returns** a list of available metrics.

**property** `output_shape` (*self*) → tuple

**Returns** output shape of a single sample of the task.

**property** `output_dtype` (*self*)

**Returns** output data type for specific task.

`matchzoo.engine.hyper_spaces`

Hyper parameter search spaces wrapping *hyperopt*.

## Module Contents

### Classes

<code>HyperoptProxy</code>	Hyperopt proxy class.
<code>choice</code>	<code>hyperopt.hp.choice()</code> proxy.
<code>quniform</code>	<code>hyperopt.hp.quniform()</code> proxy.
<code>uniform</code>	<code>hyperopt.hp.uniform()</code> proxy.

### Functions

<code>_wrap_as_composite_func</code> ( <i>self</i> , <i>other</i> , <i>func</i> )	
<code>sample</code> ( <i>space</i> )	Take a sample in the hyper space.

**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)
{'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)
{'mlp_num_layers': 8.0, 'mlp_num_units': 292.0}
```

**convert** (*self*, *name*: *str*) → hyperopt.pyll.Apply  
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)
{'mlp_num_fan_out': ...}
```

## matchzoo.engine.param

Parameter class.

## Module Contents

### Classes

---

*Param*

Parameter class.

---

matchzoo.engine.param.**SpaceType**

```
class matchzoo.engine.param.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
<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
<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'>
```

**property name** (*self*) → str

**Returns** Name of the parameter.

**property value** (*self*) → typing.Any

**Returns** Value of the parameter.

**property hyper\_space** (*self*) → *SpaceType*

**Returns** Hyper space of the parameter.

**property validator** (*self*) → typing.Callable[[typing.Any], bool]

**Returns** Validator of the parameter.

**property desc** (*self*) → str

**Returns** Parameter description.

**\_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

### Classes

---

*ParamTable*

Parameter table class.

---

**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.
```

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

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

**get** (*self*, *key*) → Param

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

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

Set *key* to parameter *param*.

**property hyper\_space** (*self*) → dict

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

**to\_frame** (*self*) → pd.DataFrame

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) → typing.Any

**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*) → typing.Iterator

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

**completed** (*self*, *exclude*: typing.Optional[list] = None) → bool

Check if all params are filled.

**Parameters** **exclude** – List of names of parameters that was excluded from being computed.

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

## Example

```
>>> import matchzoo
>>> model = matchzoo.models.DenseBaseline()
>>> model.params.completed(
...     exclude=['task', 'out_activation_func', 'embedding',
...             'embedding_input_dim', 'embedding_output_dim']
... )
True
```

**keys** (*self*) → collections.abc.KeysView

**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

#### Classes

---

*RankCrossEntropyLoss*

Creates a criterion that measures rank cross entropy loss.

---

**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']`

**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.

**property num\_neg** (*self*)

*num\_neg* getter.

## `matchzoo.losses.rank_hinge_loss`

The rank hinge loss.

## Module Contents

### Classes

---

`RankHingeLoss`

Creates a criterion that measures rank hinge loss.

---

**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']`

**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.



**property num\_neg** (*self*)  
*num\_neg* getter.

**property margin** (*self*)  
*margin* getter.

## Package Contents

### Classes

<i>RankCrossEntropyLoss</i>	Creates a criterion that measures rank cross entropy loss.
<i>RankHingeLoss</i>	Creates a criterion that measures rank hinge loss.

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

Bases: torch.nn.Module

Creates a criterion that measures rank cross entropy loss.

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

**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.

**property num\_neg** (*self*)  
*num\_neg* getter.

**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']

**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.

**property num\_neg** (*self*)  
*num\_neg* getter.

**property margin** (*self*)  
*margin* getter.

`matchzoo.metrics`

## Submodules

`matchzoo.metrics.accuracy`

Accuracy metric for Classification.

## Module Contents

### Classes

---

[\*Accuracy\*](#)

Accuracy metric.

---

**class** `matchzoo.metrics.accuracy.Accuracy`

Bases: `matchzoo.engine.base_metric.ClassificationMetric`

Accuracy metric.

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

**\_\_repr\_\_** (*self*) → str

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

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

### Classes

---

*AveragePrecision*

Average precision metric.

---

**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*) → str

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

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

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

### Classes

---

<i>CrossEntropy</i>	Cross entropy metric.
---------------------	-----------------------

---

**class** matchzoo.metrics.cross\_entropy.**CrossEntropy**  
Bases: *matchzoo.engine.base\_metric.ClassificationMetric*  
Cross entropy metric.  
**ALIAS** = ['cross\_entropy', 'ce']  
**\_\_repr\_\_**(self) → str  
    **Returns** Formated string representation of the metric.  
**\_\_call\_\_**(self, y\_true: np.array, y\_pred: np.array, eps: float = 1e-12) → float  
    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.discounted\_cumulative\_gain**

Discounted cumulative gain metric for ranking.

## Module Contents

### Classes

---

<i>DiscountedCumulativeGain</i>	Disconunted cumulative gain metric.
---------------------------------	-------------------------------------

---

```

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

```

Bases: `matchzoo.engine.base_metric.RankingMetric`

Discounted cumulative gain metric.

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

**\_\_repr\_\_** (*self*) → str

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

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

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

### Classes

---

<i>MeanAveragePrecision</i>	Mean average precision metric.
-----------------------------	--------------------------------

---

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

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

Mean average precision metric.

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

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

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

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

### Classes

---

<i>MeanReciprocalRank</i>	Mean reciprocal rank metric.
---------------------------	------------------------------

---

```
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) → str

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

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

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

### Classes

---

<i>NormalizedDiscountedCumulativeGain</i>	Normalized discounted cumulative gain metric.
---	---

---

**class** matchzoo.metrics.normalized\_discounted\_cumulative\_gain.**NormalizedDiscountedCumulative**

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

Normalized discounted cumulative gain metric.

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

**\_\_repr\_\_**(self) → str

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

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

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

#### Classes

---

*Precision*

Precision metric.

---

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

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

Precision metric.

**ALIAS = precision**

**\_\_repr\_\_** (*self*) → str

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

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

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

### Classes

<i>Precision</i>	Precision metric.
<i>DiscountedCumulativeGain</i>	Disconuted cumulative gain metric.
<i>MeanReciprocalRank</i>	Mean reciprocal rank metric.
<i>MeanAveragePrecision</i>	Mean average precision metric.
<i>NormalizedDiscountedCumulativeGain</i>	Normalized discounted cumulative gain metric.
<i>Accuracy</i>	Accuracy metric.
<i>CrossEntropy</i>	Cross entropy metric.

### Functions

---

*list\_available()* → list

---

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

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

Precision metric.

**ALIAS = precision**

**\_\_repr\_\_** (*self*) → str

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

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

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

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

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*) → str

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

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

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** Formatted string representation of the metric.

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

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, threshold:  
                                                    float = 0.0)
```

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

Normalized discounted cumulative gain metric.

```
ALIAS = ['normalized_discounted_cumulative_gain', 'ndcg']
```

```
__repr__ (self) → str
```

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

```
__call__ (self, y_true: np.array, y_pred: np.array) → float
```

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) → str
```

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

```
__call__ (self, y_true: np.array, y_pred: np.array) → float
```

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*) → str

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

**\_\_call\_\_** (*self*, *y\_true*: np.array, *y\_pred*: np.array, *eps*: float = 1e-12) → float  
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.anmm`

An implementation of aNMM Model.

## Module Contents

### Classes

---

*aNMM*

aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model.

---

**class** `matchzoo.models.anmm.aNMM` (*params*: *typing.Optional[ParamTable]* = None)

Bases: `matchzoo.engine.base_model.BaseModel`

aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model.

### Examples

```
>>> model = aNMM()
>>> model.params['embedding_output_dim'] = 300
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*) → ParamTable

**Returns** model default parameters.

**build** (*self*)

Build model structure.

aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model.

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

Forward.

`matchzoo.models.arci`

An implementation of ArcI Model.

## Module Contents

### Classes

---

*ArcI*

ArcI Model.

---

**class** matchzoo.models.arci.**ArcI** (*params: typing.Optional[ParamTable] = None*)

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*) → ParamTable

**Returns** model default parameters.

**classmethod** **get\_default\_padding\_callback** (*cls*, *fixed\_length\_left: int = 10*, *fixed\_length\_right: int = 100*, *pad\_word\_value: typing.Union[int, str] = 0*, *pad\_word\_mode: str = 'pre'*, *with\_ngram: bool = False*, *fixed\_ngram\_length: int = None*, *pad\_ngram\_value: typing.Union[int, str] = 0*, *pad\_ngram\_mode: str = 'pre'*) → BaseCallback

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 structure.

ArcI use Siamese architecture.

**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) → nn.Module
```

Make conv pool block.

## `matchzoo.models.arci`

An implementation of ArcII Model.

## Module Contents

### Classes

---

*ArcII*

ArcII Model.

---

```
class matchzoo.models.arci.ArcII (params: typing.Optional[ParamTable] = None)  
    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) → ParamTable
```

**Returns** model default parameters.

```
classmethod get_default_padding_callback (cls, fixed_length_left: int = 10, fixed_length_right: int = 100, pad_word_value: typing.Union[int, str] = 0, pad_word_mode: str = 'pre', with_ngram: bool = False, fixed_ngram_length: int = None, pad_ngram_value: typing.Union[int, str] = 0, pad_ngram_mode: str = 'pre') → BaseCallback
```

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 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*) → *nn.Module*

Make conv pool block.

## matchzoo.models.bert

An implementation of Bert Model.

## Module Contents

### Classes

---

*Bert*

Bert Model.

---

**class** `matchzoo.models.bert.Bert` (*params*: *typing.Optional[ParamTable]* = *None*)

Bases: `matchzoo.engine.base_model.BaseModel`

Bert Model.

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

**Returns** model default parameters.

**classmethod** **get\_default\_preprocessor** (*cls*, *mode*: *str* = *'bert-base-uncased'*) → *BasePreprocessor*

**Returns** Default preprocessor.

**classmethod** **get\_default\_padding\_callback** (*cls*, *fixed\_length\_left*: *int* = *None*, *fixed\_length\_right*: *int* = *None*, *pad\_value*: *typing.Union[int, str]* = *0*, *pad\_mode*: *str* = *'pre'*)

**Returns** Default padding callback.

**build** (*self*)

Build model structure.

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

Forward.

`matchzoo.models.bimpm`

An implementation of BiMPM Model.

## Module Contents

### Classes

---

<code>BiMPM</code>	BiMPM Model.
--------------------	--------------

---

### Functions

---

<code>mp_matching_func(v1, v2, w)</code>	Basic mp_matching_func.
<code>mp_matching_func_pairwise(v1, v2, w)</code>	Basic mp_matching_func_pairwise.
<code>attention(v1, v2)</code>	Attention.
<code>div_with_small_value(n, d, eps=1e-08)</code>	Small values are replaced by 1e-8 to prevent it from exploding.

---

**class** `matchzoo.models.bimpm.BiMPM` (*params*: `typing.Optional[ParamTable]` = None)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*) → `ParamTable`**Returns** model default parameters.**build** (*self*)

Make function layers.

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

Forward.

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

Init Parameters.

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

Dropout Layer.

`matchzoo.models.bimpm.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.bimpm.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.bimpm.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

### Classes

<i>CDSSM</i>	CDSSM Model implementation.
<i>Squeeze</i>	Squeeze.

**class** `matchzoo.models.cdssm.CDSSM` (*params: typing.Optional[ParamTable] = None*)  
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*) → ParamTable

**Returns** model default parameters.

**classmethod** `get_default_preprocessor` (*cls*, *truncated\_mode*: str = 'pre', *truncated\_length\_left*: typing.Optional[int] = None, *truncated\_length\_right*: typing.Optional[int] = None, *filter\_mode*: str = 'df', *filter\_low\_freq*: float = 1, *filter\_high\_freq*: float = float('inf'), *remove\_stop\_words*: bool = False, *ngram\_size*: typing.Optional[int] = 3) → BasePreprocessor

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*, *fixed\_length\_left*: int = None, *fixed\_length\_right*: int = None, *pad\_word\_value*: typing.Union[int, str] = 0, *pad\_word\_mode*: str = 'pre', *with\_ngram*: bool = True, *fixed\_ngram\_length*: int = None, *pad\_ngram\_value*: typing.Union[int, str] = 0, *pad\_ngram\_mode*: str = 'pre') → BaseCallback

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.

**\_create\_base\_network** (*self*) → nn.Module

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

### Classes

---

*ConvKNRM*

ConvKNRM Model.

---

```
class matchzoo.models.conv_knrm.ConvKNRM (params: typing.Optional[ParamTable] = None)
    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) → ParamTable
```

**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

#### Classes

---

<i>DenseBaseline</i>	A simple densely connected baseline model.
----------------------	--

---

**class** `matchzoo.models.dense_baseline.DenseBaseline` (*params:* `typing.Optional[ParamTable]` = `None`)

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*) → `ParamTable`

**Returns** model default parameters.

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

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

## `matchzoo.models.diin`

An implementation of DIIN Model.

### Module Contents

#### Classes

---

<i>DIIN</i>	DIIN model.
-------------	-------------

---

**class** `matchzoo.models.diin.DIIN` (*params:* `typing.Optional[ParamTable]` = `None`)  
Bases: `matchzoo.engine.base_model.BaseModel`

DIIN model.

## Examples

```
>>> model = DIIN()
>>> model.params['embedding_input_dim'] = 10000
>>> model.params['embedding_output_dim'] = 300
>>> model.params['mask_value'] = 0
>>> model.params['char_embedding_input_dim'] = 100
>>> model.params['char_embedding_output_dim'] = 8
>>> model.params['char_conv_filters'] = 100
>>> model.params['char_conv_kernel_size'] = 5
>>> model.params['first_scale_down_ratio'] = 0.3
>>> model.params['nb_dense_blocks'] = 3
>>> model.params['layers_per_dense_block'] = 8
>>> model.params['growth_rate'] = 20
>>> model.params['transition_scale_down_ratio'] = 0.5
>>> model.params['conv_kernel_size'] = (3, 3)
>>> model.params['pool_kernel_size'] = (2, 2)
>>> model.params['dropout_rate'] = 0.2
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*) → ParamTable

**Returns** model default parameters.

**classmethod** `get_default_preprocessor` (*cls*, *truncated\_mode*: str = 'pre', *truncated\_length\_left*: typing.Optional[int] = None, *truncated\_length\_right*: typing.Optional[int] = None, *filter\_mode*: str = 'df', *filter\_low\_freq*: float = 1, *filter\_high\_freq*: float = float('inf'), *remove\_stop\_words*: bool = False, *ngram\_size*: typing.Optional[int] = 1) → BasePreprocessor

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*, *fixed\_length\_left*: int = 10, *fixed\_length\_right*: int = 30, *pad\_word\_value*: typing.Union[int, str] = 0, *pad\_word\_mode*: str = 'pre', *with\_ngram*: bool = True, *fixed\_ngram\_length*: int = None, *pad\_ngram\_value*: typing.Union[int, str] = 0, *pad\_ngram\_mode*: str = 'pre') → BaseCallback

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 structure.

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

Forward.

## `matchzoo.models.drmm`

An implementation of DRMM Model.

### Module Contents

#### Classes

---

*DRMM*

DRMM Model.

---

**class** `matchzoo.models.drmm.DRMM` (*params: typing.Optional[ParamTable] = None*)  
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*) → ParamTable

**Returns** model default parameters.

**classmethod** `get_default_padding_callback` (*cls, fixed\_length\_left: int = None, fixed\_length\_right: int = None, pad\_value: typing.Union[int, str] = 0, pad\_mode: str = 'pre'*)

**Returns** Default padding callback.

**build** (*self*)

Build model structure.

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

Forward.

## `matchzoo.models.drmm_tks`

An implementation of DRMMTKS Model.



## Module Contents

### Classes

---

*DRMMTKS*

DRMMTKS Model.

---

**class** `matchzoo.models.drmmtk.DRMMTKS` (*params: typing.Optional[ParamTable] = None*)

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*) → ParamTable

**Returns** model default parameters.

**classmethod** `get_default_padding_callback` (*cls*, *fixed\_length\_left: int = 10*, *fixed\_length\_right: int = 100*, *pad\_word\_value: typing.Union[int, str] = 0*, *pad\_word\_mode: str = 'pre'*, *with\_ngram: bool = False*, *fixed\_ngram\_length: int = None*, *pad\_ngram\_value: typing.Union[int, str] = 0*, *pad\_ngram\_mode: str = 'pre'*) → BaseCallback

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 structure.

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

Forward.

`matchzoo.models.dssm`

An implementation of DSSM, Deep Structured Semantic Model.

## Module Contents

### Classes

---

*DSSM*Deep structured semantic model.

---

**class** `matchzoo.models.dssm.DSSM` (*params*: `typing.Optional[ParamTable]` = None)  
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*) → `ParamTable`

**Returns** model default parameters.

**classmethod** `get_default_preprocessor` (*cls*, *truncated\_mode*: `str` = 'pre', *truncated\_length\_left*: `typing.Optional[int]` = None, *truncated\_length\_right*: `typing.Optional[int]` = None, *filter\_mode*: `str` = 'df', *filter\_low\_freq*: `float` = 1, *filter\_high\_freq*: `float` = float('inf'), *remove\_stop\_words*: `bool` = False, *ngram\_size*: `typing.Optional[int]` = 3) → `BasePreprocessor`

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*)

**Returns** Default padding callback.

**build** (*self*)

Build model structure.

DSSM use Siamese architecture.

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

Forward.

`matchzoo.models.duet`

An implementation of DUET Model.

## Module Contents

### Classes

*DUET*

Duet Model.

**class** `matchzoo.models.duet.DUET` (*params: typing.Optional[ParamTable] = None*)  
 Bases: `matchzoo.engine.base_model.BaseModel`

Duet Model.

### Examples

```
>>> model = DUET()
>>> model.params['left_length'] = 10
>>> model.params['right_length'] = 40
>>> model.params['lm_filters'] = 300
>>> model.params['mlp_num_layers'] = 2
>>> model.params['mlp_num_units'] = 300
>>> model.params['mlp_num_fan_out'] = 300
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.params['vocab_size'] = 2000
>>> model.params['dm_filters'] = 300
>>> model.params['dm_conv_activation_func'] = 'relu'
>>> model.params['dm_kernel_size'] = 3
>>> model.params['dm_right_pool_size'] = 8
>>> model.params['dropout_rate'] = 0.5
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*) → ParamTable

**Returns** model default parameters.

**classmethod** `get_default_preprocessor` (*cls, truncated\_mode: str = 'pre', truncated\_length\_left: int = 10, truncated\_length\_right: int = 40, filter\_mode: str = 'df', filter\_low\_freq: float = 1, filter\_high\_freq: float = float('inf'), remove\_stop\_words: bool = False, ngram\_size: int = 3*)

**Returns** Default preprocessor.

```
classmethod get_default_padding_callback (cls, fixed_length_left: int = 10,  
                                           fixed_length_right: int = 40,  
                                           pad_word_value: typing.Union[int,  
str] = 0, pad_word_mode: str =  
'pre', with_ngram: bool = True,  
                                           fixed_ngram_length: int = None,  
                                           pad_ngram_value: typing.Union[int,  
str] = 0, pad_ngram_mode: str = 'pre') →  
BaseCallback
```

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.

```
classmethod _xor_match (cls, x, y)  
Xor match of two inputs.
```

```
build (self)  
Build model structure.
```

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

## `matchzoo.models.esim`

An implementation of ESIM Model.

## Module Contents

### Classes

---

*ESIM*

ESIM Model.

---

```
class matchzoo.models.esim.ESIM (params: typing.Optional[ParamTable] = None)  
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) → ParamTable
```

**Returns** model default parameters.

```
build (self)  
Instantiating layers.
```

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

**matchzoo.models.hbmp**

An implementation of HBMP Model.

**Module Contents****Classes***HBMP*

HBMP model.

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

HBMP model.

**Examples**

```
>>> model = HBMP ()
>>> model.params['embedding_input_dim'] = 200
>>> model.params['embedding_output_dim'] = 100
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 10
>>> model.params['mlp_num_fan_out'] = 10
>>> model.params['mlp_activation_func'] = nn.LeakyReLU(0.1)
>>> model.params['lstm_hidden_size'] = 5
>>> model.params['lstm_num'] = 3
>>> model.params['num_layers'] = 3
>>> model.params['dropout_rate'] = 0.1
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

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

**Returns** model default parameters.

**build** (*self*)

Build model structure.

HBMP use Siamese architecture.

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

Forward.

**matchzoo.models.knrm**

An implementation of KNRM Model.

## Module Contents

### Classes

---

*KNRM*

---

KNRM Model.

---

**class** `matchzoo.models.knrm.KNRM` (*params*: *typing.Optional[ParamTable]* = None)

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*) → *ParamTable*

Returns model default parameters.

**build** (*self*)

Build model structure.

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

Forward.

### `matchzoo.models.match_pyramid`

An implementation of MatchPyramid Model.

## Module Contents

### Classes

---

*MatchPyramid*

---

MatchPyramid Model.

---

**class** `matchzoo.models.match_pyramid.MatchPyramid` (*params*: *typing.Optional[ParamTable]* = None)

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

MatchPyramid Model.

## Examples

```
>>> model = MatchPyramid()
>>> model.params['embedding_output_dim'] = 300
>>> model.params['kernel_count'] = [16, 32]
>>> model.params['kernel_size'] = [[3, 3], [3, 3]]
>>> model.params['dpool_size'] = [3, 10]
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*) → ParamTable

Returns model default parameters.

**build** (*self*)

Build model structure.

MatchPyramid text matching as image recognition.

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

Forward.

**classmethod** `_make_conv_pool_block` (*cls, in\_channels: int, out\_channels: int, kernel\_size: tuple, activation: nn.Module*) → nn.Module

Make conv pool block.

### matchzoo.models.match\_srnn

An implementation of Match-SRNN Model.

## Module Contents

### Classes

---

*MatchSRNN*

Match-SRNN Model.

---

**class** `matchzoo.models.match_srnn.MatchSRNN` (*params: typing.Optional[ParamTable] = None*)

Bases: `matchzoo.engine.base_model.BaseModel`

Match-SRNN Model.

### Examples

```
>>> model = MatchSRNN()
>>> model.params['channels'] = 4
>>> model.params['units'] = 10
>>> model.params['dropout'] = 0.2
>>> model.params['direction'] = 'lt'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*) → ParamTable

**Returns** model default parameters.

**build** (*self*)  
Build model structure.

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

### `matchzoo.models.matchlstm`

An implementation of Match LSTM Model.

## Module Contents

### Classes

---

*MatchLSTM*

MatchLSTM Model.

---

**class** `matchzoo.models.matchlstm.MatchLSTM` (*params*: *typing.Optional[ParamTable]* = None)

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*) → ParamTable

**Returns** model default parameters.

**build** (*self*)  
Instantiating layers.

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

### `matchzoo.models.mvlstm`

An implementation of MVLSTM Model.



## Module Contents

### Classes

---

*MVLSTM*

MVLSTM Model.

---

**class** matchzoo.models.mvlstm.**MVLSTM** (*params: typing.Optional[ParamTable] = None*)

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

MVLSTM Model.

### Examples

```
>>> model = MVLSTM()
>>> model.params['hidden_size'] = 32
>>> model.params['top_k'] = 50
>>> model.params['mlp_num_layers'] = 2
>>> model.params['mlp_num_units'] = 20
>>> model.params['mlp_num_fan_out'] = 10
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.params['dropout_rate'] = 0.0
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

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

**Returns** model default parameters.

**classmethod** **get\_default\_padding\_callback** (*cls, fixed\_length\_left: int = 10, fixed\_length\_right: int = 40, pad\_word\_value: typing.Union[int, str] = 0, pad\_word\_mode: str = 'pre', with\_ngram: bool = False, fixed\_ngram\_length: int = None, pad\_ngram\_value: typing.Union[int, str] = 0, pad\_ngram\_mode: str = 'pre'*) → BaseCallback

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 structure.

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

Forward.

## matchzoo.models.parameter\_readme\_generator

matchzoo/models/README.md generator.

## Module Contents

### Functions

---

<code><i>_generate()</i></code>
<code><i>_make_title()</i></code>
<code><i>_make_model_class_subtitle(model_class)</i></code>
<code><i>_make_doc_section_subsubtitle()</i></code>
<code><i>_make_params_section_subsubtitle()</i></code>
<code><i>_make_model_doc(model_class)</i></code>
<code><i>_make_model_params_table(model)</i></code>
<code><i>_write_to_files(full)</i></code>

---

matchzoo.models.parameter\_readme\_generator.**\_generate**()

matchzoo.models.parameter\_readme\_generator.**\_make\_title**()

matchzoo.models.parameter\_readme\_generator.**\_make\_model\_class\_subtitle**(*model\_class*)

matchzoo.models.parameter\_readme\_generator.**\_make\_doc\_section\_subsubtitle**()

matchzoo.models.parameter\_readme\_generator.**\_make\_params\_section\_subsubtitle**()

matchzoo.models.parameter\_readme\_generator.**\_make\_model\_doc**(*model\_class*)

matchzoo.models.parameter\_readme\_generator.**\_make\_model\_params\_table**(*model*)

matchzoo.models.parameter\_readme\_generator.**\_write\_to\_files**(*full*)

## Package Contents

### Classes

<code><i>DenseBaseline</i></code>	A simple densely connected baseline model.
<code><i>DSSM</i></code>	Deep structured semantic model.
<code><i>CDSSM</i></code>	CDSSM Model implementation.
<code><i>DRMM</i></code>	DRMM Model.
<code><i>DRMMTKS</i></code>	DRMMTKS Model.
<code><i>ESIM</i></code>	ESIM Model.
<code><i>KNRM</i></code>	KNRM Model.
<code><i>ConvKNRM</i></code>	ConvKNRM Model.
<code><i>BiMPM</i></code>	BiMPM Model.
<code><i>MatchLSTM</i></code>	MatchLSTM Model.
<code><i>ArcI</i></code>	ArcI Model.
<code><i>ArcII</i></code>	ArcII Model.
<code><i>Bert</i></code>	Bert Model.
<code><i>MVLSTM</i></code>	MVLSTM Model.

continues on next page

Table 90 – continued from previous page

<i>MatchPyramid</i>	MatchPyramid Model.
<i>aNMM</i>	aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model.
<i>HBMP</i>	HBMP model.
<i>DUET</i>	Duet Model.
<i>DIIN</i>	DIIN model.
<i>MatchSRNN</i>	Match-SRNN Model.

## Functions

---

`list_available()` → list

---

**class** `matchzoo.models.DenseBaseline` (*params*: `typing.Optional[ParamTable]` = None)

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*) → `ParamTable`

**Returns** model default parameters.

**build** (*self*)

Build.

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

Forward.

**class** `matchzoo.models.DSSM` (*params*: `typing.Optional[ParamTable]` = None)

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*) → `ParamTable`

**Returns** model default parameters.

```
classmethod get_default_preprocessor(cls, truncated_mode: str = 'pre', truncated_length_left: typing.Optional[int] = None, truncated_length_right: typing.Optional[int] = None, filter_mode: str = 'df', filter_low_freq: float = 1, filter_high_freq: float = float('inf'), remove_stop_words: bool = False, ngram_size: typing.Optional[int] = 3) → BasePreprocessor
```

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)
```

**Returns** Default padding callback.

```
build(self)
```

Build model structure.

DSSM use Siamese architecture.

```
forward(self, inputs)
```

Forward.

```
class matchzoo.models.CDSSM(params: typing.Optional[ParamTable] = None)
```

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) → ParamTable
```

**Returns** model default parameters.

```
classmethod get_default_preprocessor(cls, truncated_mode: str = 'pre', truncated_length_left: typing.Optional[int] = None, truncated_length_right: typing.Optional[int] = None, filter_mode: str = 'df', filter_low_freq: float = 1, filter_high_freq: float = float('inf'), remove_stop_words: bool = False, ngram_size: typing.Optional[int] = 3) → BasePreprocessor
```

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, fixed_length_left: int = None,
                                           fixed_length_right: int = None,
                                           pad_word_value: typing.Union[int, str] = 0,
                                           pad_word_mode: str = 'pre',
                                           with_ngram: bool = True,
                                           fixed_ngram_length: int = None,
                                           pad_ngram_value: typing.Union[int, str] = 0,
                                           pad_ngram_mode: str = 'pre') → BaseCallback
```

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.

```
_create_base_network (self) → nn.Module
```

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 (params: typing.Optional[ParamTable] = None)
```

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) → ParamTable
```

**Returns** model default parameters.

```
classmethod get_default_padding_callback (cls, fixed_length_left: int = None,
fixed_length_right: int = None, pad_value:
typing.Union[int, str] = 0, pad_mode: str =
'pre')
```

**Returns** Default padding callback.

```
build (self)
    Build model structure.
```

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

```
class matchzoo.models.DRMMTKS (params: typing.Optional[ParamTable] = None)
    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) → ParamTable
```

**Returns** model default parameters.

```
classmethod get_default_padding_callback (cls, fixed_length_left: int = 10,
fixed_length_right: int = 100,
pad_word_value: typing.Union[int, str] = 0, pad_word_mode: str =
'pre', with_ngram: bool = False,
fixed_ngram_length: int = None,
pad_ngram_value: typing.Union[int, str] = 0, pad_ngram_mode: str = 'pre') →
BaseCallback
```

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 structure.
```

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

```
class matchzoo.models.ESIM (params: typing.Optional[ParamTable] = None)
    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)` → ParamTable

**Returns** model default parameters.

**build(self)**

Instantiating layers.

**forward(self, inputs)**

Forward.

**class** `matchzoo.models.KNRM` (*params: typing.Optional[ParamTable] = None*)

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)` → ParamTable

**Returns** model default parameters.

**build(self)**

Build model structure.

**forward(self, inputs)**

Forward.

**class** `matchzoo.models.ConvKNRM` (*params: typing.Optional[ParamTable] = None*)

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*) → ParamTable

**Returns** model default parameters.

**build** (*self*)

Build model structure.

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

Forward.

**class** `matchzoo.models.BiMPM` (*params*: typing.Optional[ParamTable] = None)

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*) → ParamTable

**Returns** model default parameters.

**build** (*self*)

Make function layers.

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

Forward.

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

Init Parameters.

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

Dropout Layer.

**class** `matchzoo.models.MatchLSTM` (*params*: typing.Optional[ParamTable] = None)

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*) → ParamTable

**Returns** model default parameters.

**build** (*self*)

Instantiating layers.



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

**class** matchzoo.models.**ArcI** (*params*: typing.Optional[ParamTable] = None)  
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*) → ParamTable

**Returns** model default parameters.

**classmethod** **get\_default\_padding\_callback** (*cls*, *fixed\_length\_left*: int = 10, *fixed\_length\_right*: int = 100, *pad\_word\_value*: typing.Union[int, str] = 0, *pad\_word\_mode*: str = 'pre', *with\_ngram*: bool = False, *fixed\_ngram\_length*: int = None, *pad\_ngram\_value*: typing.Union[int, str] = 0, *pad\_ngram\_mode*: str = 'pre') → BaseCallback

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 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) → nn.Module

Make conv pool block.

**class** matchzoo.models.**ArcII** (*params*: typing.Optional[ParamTable] = None)  
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*) → ParamTable

**Returns** model default parameters.

**classmethod** `get_default_padding_callback` (*cls*, *fixed\_length\_left*: int = 10, *fixed\_length\_right*: int = 100, *pad\_word\_value*: typing.Union[int, str] = 0, *pad\_word\_mode*: str = 'pre', *with\_ngram*: bool = False, *fixed\_ngram\_length*: int = None, *pad\_ngram\_value*: typing.Union[int, str] = 0, *pad\_ngram\_mode*: str = 'pre') → BaseCallback

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 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) → nn.Module

Make conv pool block.

**class** `matchzoo.models.Bert` (*params*: typing.Optional[ParamTable] = None)

Bases: `matchzoo.engine.base_model.BaseModel`

Bert Model.

**classmethod** `get_default_params` (*cls*) → ParamTable

**Returns** model default parameters.

**classmethod** `get_default_preprocessor` (*cls*, *mode*: str = 'bert-base-uncased') → BasePreprocessor

**Returns** Default preprocessor.

```
classmethod get_default_padding_callback (cls, fixed_length_left: int = None,
fixed_length_right: int = None, pad_value:
typing.Union[int, str] = 0, pad_mode: str =
'pre')
```

**Returns** Default padding callback.

```
build (self)
    Build model structure.
```

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

```
class matchzoo.models.MVLSTM (params: typing.Optional[ParamTable] = None)
```

Bases: `matchzoo.engine.base_model.BaseModel`

MVLSTM Model.

## Examples

```
>>> model = MVLSTM()
>>> model.params['hidden_size'] = 32
>>> model.params['top_k'] = 50
>>> model.params['mlp_num_layers'] = 2
>>> model.params['mlp_num_units'] = 20
>>> model.params['mlp_num_fan_out'] = 10
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.params['dropout_rate'] = 0.0
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

```
classmethod get_default_params (cls) → ParamTable
```

**Returns** model default parameters.

```
classmethod get_default_padding_callback (cls, fixed_length_left: int = 10,
fixed_length_right: int = 40,
pad_word_value: typing.Union[int,
str] = 0, pad_word_mode: str =
'pre', with_ngram: bool = False,
fixed_ngram_length: int = None,
pad_ngram_value: typing.Union[int,
str] = 0, pad_ngram_mode: str = 'pre') →
BaseCallback
```

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 structure.
```

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

```
class matchzoo.models.MatchPyramid (params: typing.Optional[ParamTable] = None)
```

Bases: `matchzoo.engine.base_model.BaseModel`

MatchPyramid Model.

## Examples

```
>>> model = MatchPyramid()
>>> model.params['embedding_output_dim'] = 300
>>> model.params['kernel_count'] = [16, 32]
>>> model.params['kernel_size'] = [[3, 3], [3, 3]]
>>> model.params['dpool_size'] = [3, 10]
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*) → ParamTable

**Returns** model default parameters.

**build** (*self*)

Build model structure.

MatchPyramid text matching as image recognition.

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

Forward.

**classmethod** `_make_conv_pool_block` (*cls, in\_channels: int, out\_channels: int, kernel\_size: tuple, activation: nn.Module*) → nn.Module

Make conv pool block.

**class** `matchzoo.models.aNMM` (*params: typing.Optional[ParamTable] = None*)

Bases: `matchzoo.engine.base_model.BaseModel`

aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model.

## Examples

```
>>> model = aNMM()
>>> model.params['embedding_output_dim'] = 300
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*) → ParamTable

**Returns** model default parameters.

**build** (*self*)

Build model structure.

aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model.

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

Forward.

**class** `matchzoo.models.HBMP` (*params: typing.Optional[ParamTable] = None*)

Bases: `matchzoo.engine.base_model.BaseModel`

HBMP model.

## Examples

```
>>> model = HBMP()
>>> model.params['embedding_input_dim'] = 200
>>> model.params['embedding_output_dim'] = 100
>>> model.params['mlp_num_layers'] = 1
>>> model.params['mlp_num_units'] = 10
>>> model.params['mlp_num_fan_out'] = 10
>>> model.params['mlp_activation_func'] = nn.LeakyReLU(0.1)
>>> model.params['lstm_hidden_size'] = 5
>>> model.params['lstm_num'] = 3
>>> model.params['num_layers'] = 3
>>> model.params['dropout_rate'] = 0.1
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)` → ParamTable

**Returns** model default parameters.

**build(self)**

Build model structure.

HBMP use Siamese architecture.

**forward(self, inputs)**

Forward.

**class** `matchzoo.models.DUET` (*params: typing.Optional[ParamTable] = None*)

Bases: `matchzoo.engine.base_model.BaseModel`

Duet Model.

## Examples

```
>>> model = DUET()
>>> model.params['left_length'] = 10
>>> model.params['right_length'] = 40
>>> model.params['lm_filters'] = 300
>>> model.params['mlp_num_layers'] = 2
>>> model.params['mlp_num_units'] = 300
>>> model.params['mlp_num_fan_out'] = 300
>>> model.params['mlp_activation_func'] = 'relu'
>>> model.params['vocab_size'] = 2000
>>> model.params['dm_filters'] = 300
>>> model.params['dm_conv_activation_func'] = 'relu'
>>> model.params['dm_kernel_size'] = 3
>>> model.params['dm_right_pool_size'] = 8
>>> model.params['dropout_rate'] = 0.5
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params(cls)` → ParamTable

**Returns** model default parameters.

```
classmethod get_default_preprocessor(cls, truncated_mode: str = 'pre', truncated_length_left: int = 10, truncated_length_right: int = 40, filter_mode: str = 'df', filter_low_freq: float = 1, filter_high_freq: float = float('inf'), remove_stop_words: bool = False, ngram_size: int = 3)
```

**Returns** Default preprocessor.

```
classmethod get_default_padding_callback(cls, fixed_length_left: int = 10, fixed_length_right: int = 40, pad_word_value: typing.Union[int, str] = 0, pad_word_mode: str = 'pre', with_ngram: bool = True, fixed_ngram_length: int = None, pad_ngram_value: typing.Union[int, str] = 0, pad_ngram_mode: str = 'pre') → BaseCallback
```

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.

```
classmethod _xor_match(cls, x, y)
Xor match of two inputs.
```

```
build(self)
Build model structure.
```

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

```
class matchzoo.models.DIIN(params: typing.Optional[ParamTable] = None)
Bases: matchzoo.engine.base_model.BaseModel
```

DIIN model.

## Examples

```
>>> model = DIIN()
>>> model.params['embedding_input_dim'] = 10000
>>> model.params['embedding_output_dim'] = 300
>>> model.params['mask_value'] = 0
>>> model.params['char_embedding_input_dim'] = 100
>>> model.params['char_embedding_output_dim'] = 8
>>> model.params['char_conv_filters'] = 100
>>> model.params['char_conv_kernel_size'] = 5
>>> model.params['first_scale_down_ratio'] = 0.3
>>> model.params['nb_dense_blocks'] = 3
>>> model.params['layers_per_dense_block'] = 8
>>> model.params['growth_rate'] = 20
>>> model.params['transition_scale_down_ratio'] = 0.5
>>> model.params['conv_kernel_size'] = (3, 3)
>>> model.params['pool_kernel_size'] = (2, 2)
>>> model.params['dropout_rate'] = 0.2
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*) → ParamTable

**Returns** model default parameters.

**classmethod** `get_default_preprocessor` (*cls*, *truncated\_mode*: str = 'pre', *truncated\_length\_left*: typing.Optional[int] = None, *truncated\_length\_right*: typing.Optional[int] = None, *filter\_mode*: str = 'df', *filter\_low\_freq*: float = 1, *filter\_high\_freq*: float = float('inf'), *remove\_stop\_words*: bool = False, *ngram\_size*: typing.Optional[int] = 1) → BasePreprocessor

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*, *fixed\_length\_left*: int = 10, *fixed\_length\_right*: int = 30, *pad\_word\_value*: typing.Union[int, str] = 0, *pad\_word\_mode*: str = 'pre', *with\_ngram*: bool = True, *fixed\_ngram\_length*: int = None, *pad\_ngram\_value*: typing.Union[int, str] = 0, *pad\_ngram\_mode*: str = 'pre') → BaseCallback

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 structure.

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

Forward.

**class** `matchzoo.models.MatchSRNN` (*params*: typing.Optional[ParamTable] = None)

Bases: `matchzoo.engine.base_model.BaseModel`

Match-SRNN Model.

## Examples

```
>>> model = MatchSRNN()
>>> model.params['channels'] = 4
>>> model.params['units'] = 10
>>> model.params['dropout'] = 0.2
>>> model.params['direction'] = 'lt'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

**classmethod** `get_default_params` (*cls*) → ParamTable

**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

#### Classes

<i>Attention</i>	Attention module.
<i>BidirectionalAttention</i>	Computing the soft attention between two sequence.
<i>MatchModule</i>	Computing the match representation for Match LSTM.

**class** `matchzoo.modules.attention.Attention` (*input\_size*: int = 100)

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])
>>> x_mask = torch.BoolTensor(4, 5)
>>> attention(x, x_mask).shape
torch.Size([4, 5])
```

**forward** (*self*, *x*, *x\_mask*)  
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

#### Classes

---

*BertModule*

Bert module.

---

**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.character\_embedding

Character embedding module.

### Module Contents

#### Classes

---

*CharacterEmbedding*

Character embedding module.

---

```
class matchzoo.modules.character_embedding.CharacterEmbedding(char_embedding_input_dim:
    int = 100,
    char_embedding_output_dim:
    int = 8,
    char_conv_filters:
    int = 100,
    char_conv_kernel_size:
    int = 5)
```

Bases: torch.nn.Module

Character embedding module.

#### Parameters

- **char\_embedding\_input\_dim** – The input dimension of character embedding layer.
- **char\_embedding\_output\_dim** – The output dimension of character embedding layer.
- **char\_conv\_filters** – The filter size of character convolution layer.
- **char\_conv\_kernel\_size** – The kernel size of character convolution layer.

#### Examples

```
>>> import torch
>>> character_embedding = CharacterEmbedding()
>>> x = torch.ones(10, 32, 16, dtype=torch.long)
>>> x.shape
torch.Size([10, 32, 16])
>>> character_embedding(x).shape
torch.Size([10, 32, 100])
```

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

`matchzoo.modules.dense_net`

DenseNet module.

## Module Contents

### Classes

<i>DenseBlock</i>	Dense block of DenseNet.
<i>DenseNet</i>	DenseNet module.

```
class matchzoo.modules.dense_net.DenseBlock (in_channels, growth_rate: int =
    20, kernel_size: tuple = 2, 2, layers_per_dense_block: int = 3)
```

Bases: `torch.nn.Module`

Dense block of DenseNet.

```
forward (self, x)
    Forward.
```

```
classmethod _make_conv_block (cls, in_channels: int, out_channels: int, kernel_size: tuple)
    → nn.Module
    Make conv block.
```

```
class matchzoo.modules.dense_net.DenseNet (in_channels, nb_dense_blocks: int = 3, layers_per_dense_block:
int = 3, growth_rate: int = 10, transition_scale_down_ratio: float = 0.5, conv_kernel_size: tuple = 2, 2,
pool_kernel_size: tuple = 2, 2)
```

Bases: `torch.nn.Module`

DenseNet module.

#### Parameters

- **in\_channels** – Feature size of input.
- **nb\_dense\_blocks** – The number of blocks in densenet.
- **layers\_per\_dense\_block** – The number of convolution layers in dense block.
- **growth\_rate** – The filter size of each convolution layer in dense block.
- **transition\_scale\_down\_ratio** – The channel scale down ratio of the convolution layer in transition block.
- **conv\_kernel\_size** – The kernel size of convolution layer in dense block.
- **pool\_kernel\_size** – The kernel size of pooling layer in transition block.

```
property out_channels (self) → int
    out_channels getter.
```

```
forward (self, x)
    Forward.
```

```
classmethod _make_transition_block (cls, in_channels: int, transition_scale_down_ratio:
float, pool_kernel_size: tuple) → nn.Module
```

`matchzoo.modules.dropout`

## Module Contents

### Classes

---

*RNNDropout*

Dropout for RNN.

---

**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

### Classes

---

*GaussianKernel*

Gaussian kernel module.

---

**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

### Classes

---

<i>Matching</i>	Module that computes a matching matrix between samples in two tensors.
-----------------	--

---

**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.matching_tensor`

Matching Tensor module.

## Module Contents

### Classes

---

<code>MatchingTensor</code>	Module that captures the basic interactions between two tensors.
-----------------------------	--

---

**class** `matchzoo.modules.matching_tensor.MatchingTensor` (*matching\_dim: int, channels: int = 4, normalize: bool = True, init\_diag: bool = True*)

Bases: `torch.nn.Module`

Module that captures the basic interactions between two tensors.

#### Parameters

- **matching\_dims** – Word dimension of two interaction texts.
- **channels** – Number of word interaction tensor channels.
- **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.
- **init\_diag** – Whether to initialize the diagonal elements of the matrix.

#### Examples

```
>>> import matchzoo as mz
>>> matching_dim = 5
>>> matching_tensor = mz.modules.MatchingTensor(
...     matching_dim,
...     channels=4,
...     normalize=True,
...     init_diag=True
... )
```

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

The computation logic of `MatchingTensor`.

**Parameters** **inputs** – two input tensors.

`matchzoo.modules.semantic_composite`

Semantic composite module for DIIN model.

## Module Contents

### Classes

---

*SemanticComposite*SemanticComposite module.

---

```
class matchzoo.modules.semantic_composite.SemanticComposite(in_features,
                                                         dropout_rate: float
                                                         = 0.0)
```

Bases: torch.nn.Module

SemanticComposite module.

Apply a self-attention layer and a semantic composite fuse gate to compute the encoding result of one tensor.

#### Parameters

- **in\_features** – Feature size of input.
- **dropout\_rate** – The dropout rate.

#### Examples

```
>>> import torch
>>> module = SemanticComposite(in_features=10)
>>> x = torch.randn(4, 5, 10)
>>> x.shape
torch.Size([4, 5, 10])
>>> module(x).shape
torch.Size([4, 5, 10])
```

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

**matchzoo.modules.spatial\_gru**

Spatial GRU module.

## Module Contents

### Classes

---

*SpatialGRU*Spatial GRU Module.

---

```
class matchzoo.modules.spatial_gru.SpatialGRU(channels: int = 4, units: int = 10,
                                               activation: typing.Union[str, typing.Type[nn.Module], nn.Module]
                                               = 'tanh', recurrent_activation: typing.Union[str, typing.Type[nn.Module],
                                               nn.Module] = 'sigmoid', direction: str = 'lt')
```

Bases: torch.nn.Module

Spatial GRU Module.

### Parameters

- **channels** – Number of word interaction tensor channels.
- **units** – Number of SpatialGRU units.
- **activation** – Activation function to use, one of: - String: name of an activation - Torch Module subclass - Torch Module instance Default: hyperbolic tangent (*tanh*).
- **recurrent\_activation** – Activation function to use for the recurrent step, one of:
  - String: name of an activation
  - Torch Module subclass
  - Torch Module instanceDefault: sigmoid activation (*sigmoid*).
- **direction** – Scanning direction. *lt* (i.e., left top) indicates the scanning from left top to right bottom, and *rb* (i.e., right bottom) indicates the scanning from right bottom to left top.

### Examples

```
>>> import matchzoo as mz
>>> channels, units= 4, 10
>>> spatial_gru = mz.modules.SpatialGRU(channels, units)
```

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

Initialize parameters.

**softmax\_by\_row** (*self*, *z: torch.tensor*) → tuple

Conduct softmax on each dimension across the four gates.

**calculate\_recurrent\_unit** (*self*, *inputs: torch.tensor*, *states: list*, *i: int*, *j: int*)

Calculate recurrent unit.

### Parameters

- **inputs** – A tensor which contains interaction between left text and right text.
- **states** – An array of tensors which stores the hidden state of every step.
- **i** – Recurrent row index.
- **j** – Recurrent column index.

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

Perform SpatialGRU on word interaction matrix.

Parameters **inputs** – input tensors.



matchzoo.modules.stacked\_brnn

## Module Contents

### Classes

---

*StackedBRNN*

Stacked Bi-directional RNNs.

---

```
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

### Classes

<i>Attention</i>	Attention module.
<i>BidirectionalAttention</i>	Computing the soft attention between two sequence.
<i>MatchModule</i>	Computing the match representation for Match LSTM.
<i>RNNDropout</i>	Dropout for RNN.
<i>StackedBRNN</i>	Stacked Bi-directional RNNs.
<i>GaussianKernel</i>	Gaussian kernel module.
<i>Matching</i>	Module that computes a matching matrix between samples in two tensors.
<i>BertModule</i>	Bert module.
<i>CharacterEmbedding</i>	Character embedding module.
<i>SemanticComposite</i>	SemanticComposite module.
<i>DenseNet</i>	DenseNet module.
<i>MatchingTensor</i>	Module that captures the basic interactions between two tensors.
<i>SpatialGRU</i>	Spatial GRU Module.

```
class matchzoo.modules.Attention (input_size: int = 100)
```

```
    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])
>>> x_mask = torch.BoolTensor(4, 5)
>>> attention(x, x_mask).shape
torch.Size([4, 5])
```

```
forward (self, x, x_mask)
```

```
    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)
>>> 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.

```
class matchzoo.modules.CharacterEmbedding (char_embedding_input_dim: int =
                                         100, char_embedding_output_dim: int
                                         = 8, char_conv_filters: int = 100,
                                         char_conv_kernel_size: int = 5)
```

Bases: torch.nn.Module

Character embedding module.

#### Parameters

- **char\_embedding\_input\_dim** – The input dimension of character embedding layer.
- **char\_embedding\_output\_dim** – The output dimension of character embedding layer.
- **char\_conv\_filters** – The filter size of character convolution layer.
- **char\_conv\_kernel\_size** – The kernel size of character convolution layer.

#### Examples

```
>>> import torch
>>> character_embedding = CharacterEmbedding()
>>> x = torch.ones(10, 32, 16, dtype=torch.long)
>>> x.shape
torch.Size([10, 32, 16])
>>> character_embedding(x).shape
torch.Size([10, 32, 100])
```

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

```
class matchzoo.modules.SemanticComposite (in_features, dropout_rate: float = 0.0)
```

Bases: torch.nn.Module

SemanticComposite module.

Apply a self-attention layer and a semantic composite fuse gate to compute the encoding result of one tensor.

#### Parameters

- **in\_features** – Feature size of input.
- **dropout\_rate** – The dropout rate.

## Examples

```
>>> import torch
>>> module = SemanticComposite(in_features=10)
>>> x = torch.randn(4, 5, 10)
>>> x.shape
torch.Size([4, 5, 10])
>>> module(x).shape
torch.Size([4, 5, 10])
```

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

```
class matchzoo.modules.DenseNet (in_channels, nb_dense_blocks: int = 3, layers_per_dense_block: int = 3, growth_rate: int = 10, transition_scale_down_ratio: float = 0.5, conv_kernel_size: tuple = 2, 2, pool_kernel_size: tuple = 2, 2)
```

Bases: torch.nn.Module

DenseNet module.

### Parameters

- **in\_channels** – Feature size of input.
- **nb\_dense\_blocks** – The number of blocks in densenet.
- **layers\_per\_dense\_block** – The number of convolution layers in dense block.
- **growth\_rate** – The filter size of each convolution layer in dense block.
- **transition\_scale\_down\_ratio** – The channel scale down ratio of the convolution layer in transition block.
- **conv\_kernel\_size** – The kernel size of convolution layer in dense block.
- **pool\_kernel\_size** – The kernel size of pooling layer in transition block.

**property** **out\_channels** (*self*) → int  
*out\_channels* getter.

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

```
classmethod _make_transition_block (cls, in_channels: int, transition_scale_down_ratio: float, pool_kernel_size: tuple) → nn.Module
```

```
class matchzoo.modules.MatchingTensor (matching_dim: int, channels: int = 4, normalize: bool = True, init_diag: bool = True)
```

Bases: torch.nn.Module

Module that captures the basic interactions between two tensors.

### Parameters

- **matching\_dims** – Word dimension of two interaction texts.
- **channels** – Number of word interaction tensor channels.
- **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.
- **init\_diag** – Whether to initialize the diagonal elements of the matrix.

## Examples

```
>>> import matchzoo as mz
>>> matching_dim = 5
>>> matching_tensor = mz.modules.MatchingTensor(
...     matching_dim,
...     channels=4,
...     normalize=True,
...     init_diag=True
... )
```

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

The computation logic of MatchingTensor.

**Parameters** **inputs** – two input tensors.

```
class matchzoo.modules.SpatialGRU (channels: int = 4, units: int = 10, activation: typing.Union[str, typing.Type[nn.Module], nn.Module] = 'tanh', recurrent_activation: typing.Union[str, typing.Type[nn.Module], nn.Module] = 'sigmoid', direction: str = 'lt')
```

Bases: torch.nn.Module

Spatial GRU Module.

### Parameters

- **channels** – Number of word interaction tensor channels.
- **units** – Number of SpatialGRU units.
- **activation** – Activation function to use, one of:
  - String: name of an activation - Torch Module subclass - Torch Module instance
  - Default: hyperbolic tangent (*tanh*).
- **recurrent\_activation** – Activation function to use for the recurrent step, one of:
  - String: name of an activation
  - Torch Module subclass
  - Torch Module instance
  - Default: sigmoid activation (*sigmoid*).
- **direction** – Scanning direction. *lt* (i.e., left top) indicates the scanning from left top to right bottom, and *rb* (i.e., right bottom) indicates the scanning from right bottom to left top.

## Examples

```
>>> import matchzoo as mz
>>> channels, units= 4, 10
>>> spatial_gru = mz.modules.SpatialGRU(channels, units)
```

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

Initialize parameters.

**softmax\_by\_row** (*self*, *z*: torch.tensor) → tuple

Conduct softmax on each dimension across the four gates.

**calculate\_recurrent\_unit** (*self*, *inputs*: torch.tensor, *states*: list, *i*: int, *j*: int)

Calculate recurrent unit.

### Parameters

- **inputs** – A tensor which contains interaction between left text and right text.
- **states** – An array of tensors which stores the hidden state of every step.
- **i** – Recurrent row index.
- **j** – Recurrent column index.

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

Perform SpatialGRU on word interaction matrix.

Parameters **inputs** – input tensors.

`matchzoo.preprocessors`

### Subpackages

`matchzoo.preprocessors.units`

### Submodules

`matchzoo.preprocessors.units.character_index`

### Module Contents

### Classes

---

*CharacterIndex*

CharacterIndexUnit for DIIN model.

---

**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) → 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

### Classes

---

*DigitRemoval*

Process unit to remove digits.

---

**class** `matchzoo.preprocessors.units.digit_removal.DigitRemoval`

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit to remove digits.

**transform** (*self*, *input\_*: list) → 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

### Classes

---

*FrequencyFilter*

Frequency filter unit.

---

**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*) → *list*

Transform a list of tokens by filtering out unwanted words.

**classmethod** **\_tf** (*cls*, *list\_of\_tokens*: *list*) → *dict*

**classmethod** **\_df** (*cls*, *list\_of\_tokens*: *list*) → *dict*

**classmethod** **\_idf** (*cls*, *list\_of\_tokens*: *list*) → *dict*

**matchzoo.preprocessors.units.lemmatization****Module Contents****Classes**

---

*Lemmatization*

---

Process unit for token lemmatization.

---

**class** matchzoo.preprocessors.units.lemmatization.**Lemmatization**

Bases: *matchzoo.preprocessors.units.unit.Unit*

Process unit for token lemmatization.

**transform** (*self*, *input\_*: *list*) → *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

### Classes

---

*Lowercase*

Process unit for text lower case.

---

**class** `matchzoo.preprocessors.units.lowercase.Lowercase`

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for text lower case.

**transform** (*self*, *input\_*: list) → 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

### Classes

---

*MatchingHistogram*

MatchingHistogramUnit Class.

---

**class** `matchzoo.preprocessors.units.matching_histogram.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) → list  
Transform the input text.

## matchzoo.preprocessors.units.ngram\_letter

### Module Contents

### Classes

---

*NgramLetter*

Process unit for n-letter generation.

---

**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) → 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

### Classes

---

*PuncRemoval*

Process unit for remove punctuations.

---

**class** `matchzoo.preprocessors.units.punc_removal.PuncRemoval`

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for remove punctuations.

**\_MATCH\_PUNC**

**transform** (*self*, *input\_*: list) → list

Remove punctuations from list of tokens.

**Parameters** *input* – list of tokens.

**Return** *rv* tokens without punctuation.

`matchzoo.preprocessors.units.stateful_unit`

## Module Contents

### Classes

---

*StatefulUnit*

Unit with inner state.

---

**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*.

**property** *state* (*self*)

Get current context. Same as *unit.context*.

Deprecated since v2.2.0, and will be removed in the future. Used *unit.context* instead.

**property** *context* (*self*)

Get current context. Same as *unit.state*.

**abstract fit** (*self, input\_: typing.Any*)  
Abstract base method, need to be implemented in subclass.

`matchzoo.preprocessors.units.stemming`

### Module Contents

#### Classes

---

<i>Stemming</i>	Process unit for token stemming.
-----------------	----------------------------------

---

**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*) → 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

#### Classes

---

<i>StopRemoval</i>	Process unit to remove stop words.
--------------------	------------------------------------

---

**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'>
```

**transform** (*self, input\_: list*) → 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.

**property stopwords** (*self*) → list  
Get stopwords based on language.

**Params lang** language code.

**Returns** list of stop words.

`matchzoo.preprocessors.units.tokenize`

## Module Contents

### Classes

---

*Tokenize*

Process unit for text tokenization.

---

**class** `matchzoo.preprocessors.units.tokenize.Tokenize`

Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for text tokenization.

**transform** (*self*, *input\_*: *str*) → list

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

### Classes

---

*TruncatedLength*

TruncatedLengthUnit Class.

---

**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
```

**t transform** (*self*, *input\_*: list) → 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

### Classes

---

*Unit*

Process unit do not persive state (i.e. do not need fit).

---

**class** `matchzoo.preprocessors.units.unit.Unit`

Process unit do not persive state (i.e. do not need fit).

**abstract transform** (*self*, *input\_*: typing.Any)

Abstract base method, need to be implemented in subclass.

`matchzoo.preprocessors.units.vocabulary`

## Module Contents

### Classes

---

*Vocabulary*

Vocabulary class.

---

**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
{'[PAD]': 0, '[OOV]': 1, 'D': 2, 'A': 3, 'B': 4, 'C': 5, 'E': 6}
>>> index_term = vocab.state['index_term']
>>> index_term
{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) → list  
Transform a list of tokens to corresponding indices.

`matchzoo.preprocessors.units.word_exact_match`

## Module Contents

### Classes

---

*WordExactMatch*

WordExactUnit Class.

---

```
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)
>>> 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\_*) → list

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

#### Classes

---

*WordHashing*

Word-hashing layer for DSSM-based models.

---

**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) → 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

### Classes

<i>Unit</i>	Process unit do not persive state (i.e. do not need fit).
<i>DigitRemoval</i>	Process unit to remove digits.
<i>FrequencyFilter</i>	Frequency filter unit.
<i>Lemmatization</i>	Process unit for token lemmatization.
<i>Lowercase</i>	Process unit for text lower case.
<i>MatchingHistogram</i>	MatchingHistogramUnit Class.
<i>NgramLetter</i>	Process unit for n-letter generation.
<i>PuncRemoval</i>	Process unit for remove punctuations.
<i>StatefulUnit</i>	Unit with inner state.
<i>Stemming</i>	Process unit for token stemming.
<i>StopRemoval</i>	Process unit to remove stop words.
<i>Tokenize</i>	Process unit for text tokenization.
<i>Vocabulary</i>	Vocabulary class.
<i>WordHashing</i>	Word-hashing layer for DSSM-based models.
<i>CharacterIndex</i>	CharacterIndexUnit for DIIN model.
<i>WordExactMatch</i>	WordExactUnit Class.
<i>TruncatedLength</i>	TruncatedLengthUnit Class.

## Functions

---

```
list_available() → list
```

---

**class** `matchzoo.preprocessors.units.Unit`  
Process unit do not persive state (i.e. do not need fit).

**abstract 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*) → *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'])
>>> 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*) → *list*  
Transform a list of tokens by filtering out unwanted words.

**classmethod** **\_tf** (*cls*, *list\_of\_tokens*: *list*) → *dict*

**classmethod** **\_df** (*cls*, *list\_of\_tokens*: *list*) → *dict*

**classmethod** **\_idf** (*cls*, *list\_of\_tokens*: *list*) → *dict*

**class** `matchzoo.preprocessors.units.Lemmatization`  
Bases: `matchzoo.preprocessors.units.unit.Unit`

Process unit for token lemmatization.

**transform** (*self*, *input\_*: *list*) → *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*) → *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) → 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) → 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) → 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*.

**property state** (*self*)

Get current context. Same as *unit.context*.

Deprecated since v2.2.0, and will be removed in the future. Used *unit.context* instead.

**property context** (*self*)

Get current context. Same as *unit.state*.

**abstract 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*) → *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'>
```

**transform** (*self*, *input\_*: *list*) → *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.

**property stopwords** (*self*) → *list*

Get stopwords based on language.

**Params lang** language code.

**Returns** list of stop words.

**class** matchzoo.preprocessors.units.**Tokenize**  
 Bases: *matchzoo.preprocessors.units.unit.Unit*

Process unit for text tokenization.

**transform** (*self*, *input\_*: *str*) → list  
 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
{'[PAD]': 0, '[OOV]': 1, 'D': 2, 'A': 3, 'B': 4, 'C': 5, 'E': 6}
>>> index_term = vocab.state['index_term']
>>> index_term
{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) → 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) → 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) → 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\_*) → list

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'*)

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) → 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

### Classes

---

*BasicPreprocessor*

Baisc preprocessor helper.

---

```

class matchzoo.preprocessors.basic_preprocessor.BasicPreprocessor (truncated_mode:
                                                                    str          =
                                                                    'pre', trun-
                                                                    cated_length_left:
                                                                    int          =
                                                                    None, trun-
                                                                    cated_length_right:
                                                                    int          =
                                                                    None, fil-
                                                                    ter_mode:
                                                                    str          =
                                                                    'df', fil-
                                                                    ter_low_freq:
                                                                    float       =
                                                                    1, fil-
                                                                    ter_high_freq:
                                                                    float       =
                                                                    float('inf'),
                                                                    re-
                                                                    move_stop_words:
                                                                    bool        =
                                                                    False,
                                                                    ngram_size:
                                                                    typing.Optional[int]
                                                                    = None)

```

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,

```

(continues on next page)

(continued from previous page)

```

...     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) → *DataPack*  
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****Classes**


---

*BertPreprocessor*

Baisc preprocessor helper.

---

**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*) → *DataPack*  
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**

**Functions**

---

*build\_unit\_from\_data\_pack*(*unit*: *StatefulUnit*, *data\_pack*: *mz.DataPack*, *mode*: *str = 'both'*, *flatten*: *bool = True*, *verbose*: *int = 1*) → *StatefulUnit*

---

`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

### Functions

---

*build\_vocab\_unit*(data\_pack: DataPack, mode: str = ‘both’, verbose: int = 1) → Vocabulary given *data\_pack*.

---

`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.chain_transform`

Wrapper function organizes a number of transform functions.

## Module Contents

### Functions

---

*chain\_transform*(units: typing.List[Unit]) → typing.Callable Compose unit transformations into a single function.

---

`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.naive_preprocessor`

Naive Preprocessor.

## Module Contents

### Classes

---

*NaivePreprocessor*

Naive preprocessor.

---

**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*) → `DataPack`

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

### Classes

<i>NaivePreprocessor</i>	Naive preprocessor.
<i>BasicPreprocessor</i>	Basic preprocessor helper.
<i>BertPreprocessor</i>	Basic preprocessor helper.

### Functions

<i>list_available()</i> → list
--------------------------------

#### **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*) → DataPack

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', truncated_length_left: int = None, truncated_length_right: int = None, filter_mode: str = 'df', filter_low_freq: float = 1, filter_high_freq: float = float('inf'), remove_stop_words: bool = False, ngram_size: typing.Optional[int] = None)
```

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*) → *DataPack*  
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.BertPreprocessor` (*mode*: *str = 'bert-base-uncased'*)

Bases: `matchzoo.engine.base_preprocessor.BasePreprocessor`

Basic 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*) → *DataPack*  
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

### Classes

---

*Classification*

Classification task.

---

**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**

**property num\_classes** (*self*) → int

**Returns** number of classes to classify.

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

**Returns** a list of available losses.

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

**Returns** a list of available metrics.

**property output\_shape** (*self*) → tuple

**Returns** output shape of a single sample of the task.

**property output\_dtype** (*self*)

**Returns** target data type, expect *int* as output.

**\_\_str\_\_** (*self*)

**Returns** Task name as string.

## matchzoo.tasks.ranking

Ranking task.

## Module Contents

### Classes

---

*Ranking*

Ranking Task.

---

**class** matchzoo.tasks.ranking.**Ranking** (*losses=None, metrics=None*)

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**

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

**Returns** a list of available losses.

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

**Returns** a list of available metrics.

**property output\_shape** (*self*) → tuple

**Returns** output shape of a single sample of the task.

**property output\_dtype** (*self*)

**Returns** target data type, expect *float* as output.

**\_\_str\_\_** (*self*)

**Returns** Task name as string.

## Package Contents

### Classes

<i>Classification</i>	Classification task.
<i>Ranking</i>	Ranking Task.

**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**

**property num\_classes** (*self*) → int

**Returns** number of classes to classify.

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

**Returns** a list of available losses.

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

**Returns** a list of available metrics.

**property output\_shape** (*self*) → tuple

**Returns** output shape of a single sample of the task.

**property output\_dtype** (*self*)

**Returns** target data type, expect *int* as output.

**\_\_str\_\_** (*self*)

**Returns** Task name as string.

**class** matchzoo.tasks.**Ranking** (*losses=None, metrics=None*)

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**

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

**Returns** a list of available losses.

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

**Returns** a list of available metrics.

**property output\_shape** (*self*) → tuple

**Returns** output shape of a single sample of the task.

**property output\_dtype** (*self*)

**Returns** target data type, expect *float* as output.

**\_\_str\_\_** (*self*)

**Returns** Task name as string.

`matchzoo.trainers`

## Submodules

`matchzoo.trainers.trainer`

Base Trainer.

## Module Contents

### Classes

---

<i>Trainer</i>	MatchZoo trainer.
----------------	-------------------

---

**class** `matchzoo.trainers.trainer.Trainer` (*model: BaseModel, optimizer: optim.Optimizer, trainloader: DataLoader, validloader: DataLoader, device: typing.Union[torch.device, int, list, None] = 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, 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`, use the current device. If `torch.device` or `int`, use device specified by user. If `list`, use data parallel.
- **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 fo events to wait if no improvement and then stop the training.
- **key** – Key of metric to be compared.

- **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.Union[torch.device, int, list, None] = None*)

Load model.

#### Parameters

- **model** – `BaseModel` instance.
- **device** – The desired device of returned tensor. Default: if None, use the current device. If `torch.device` or int, use device specified by user. If list, use 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*) → *np.array*

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

### Classes

---

*Trainer*

MatchZoo trainer.

---

```
class matchzoo.trainers.Trainer(model: BaseModel, optimizer: optim.Optimizer, trainloader: DataLoader, validloader: DataLoader, device: typing.Union[torch.device, int, list, None] = 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, 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`, use the current device. If `torch.device` or `int`, use device specified by user. If `list`, use data parallel.
- **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 fo events to wait if no improvement and then stop the training.
- **key** – Key of metric to be compared.
- **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.Union[torch.device, int, list, None]* = *None*)  
Load model.

**Parameters**

- **model** – *BaseModel* instance.
- **device** – The desired device of returned tensor. Default: if *None*, use the current device. If *torch.device* or int, use device specified by user. If list, use 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*) → *np.array*

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.

## `matchzoo.utils`

### Submodules

#### `matchzoo.utils.average_meter`

Average meter.

## Module Contents

### Classes

---

*AverageMeter*

Computes and stores the average and current value.

---

**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
```

**reset** (*self*)

Reset AverageMeter.

**update** (*self*, *val*, *n=1*)

Update value.

**property avg** (*self*)

Get avg.

**matchzoo.utils.early\_stopping**

Early stopping.

## Module Contents

### Classes

---

*EarlyStopping*

EarlyStopping stops training if no improvement after a given patience.

---

**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.

**state\_dict** (*self*) → typing.Dict[str, typing.Any]  
 A *Trainer* can use this to serialize the state.

**load\_state\_dict** (*self*, *state\_dict*: typing.Dict[str, typing.Any]) → None  
 Hydrate a early stopping from a serialized state.

**update** (*self*, *result*: list)  
 Call function.

**property best\_so\_far** (*self*) → bool  
 Returns best so far.

**property is\_best\_so\_far** (*self*) → bool  
 Returns true if it is the best so far.

**property should\_stop\_early** (*self*) → bool  
 Returns true if improvement has stopped for long enough.

### matchzoo.utils.get\_file

Download file.

## Module Contents

### Classes

<i>Progbar</i>	Displays a progress bar.
----------------	--------------------------

### Functions

<i>_extract_archive</i> (file_path, path='.', archive_format='auto')	Extracts an archive if it matches tar, tar.gz, tar.bz, or zip formats.
<i>get_file</i> (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.
<i>validate_file</i> (fpath, file_hash, algorithm='auto', chunk_size=65535)	Validates a file against a sha256 or md5 hash.
<i>_hash_file</i> (fpath, algorithm='sha256', chunk_size=65535)	Calculates a file sha256 or md5 hash.

**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

### Functions

---

```
list_recursive_concrete_subclasses(base) List all concrete subclasses of base recursively.  
_filter_concrete(classes)  
_bfs(base)
```

---

```
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****Functions**


---

`one_hot`(indices: int, num\_classes: int) → np.ndarray

**return** A one-hot encoded vector.

---

matchzoo.utils.one\_hot.**one\_hot** (indices: int, num\_classes: int) → np.ndarray

**Returns** A one-hot encoded vector.

**matchzoo.utils.parse****Module Contents****Functions**


---

<code>_parse</code> (identifier: typing.Union[str, typing.Type[nn.Module], nn.ModuleDict, target: str) → nn.Module	Parse loss and activation.
<code>parse_activation</code> (identifier: typing.Union[str, typing.Type[nn.Module], nn.Module]) → nn.Module	Retrieves a torch Module instance.
<code>parse_loss</code> (identifier: typing.Union[str, typing.Type[nn.Module], nn.Module], task: typing.Optional[str] = None) → nn.Module	Retrieves a torch Module instance.
<code>_parse_metric</code> (metric: typing.Union[str, typing.Type[BaseMetric], BaseMetric], Metrix: typing.Type[BaseMetric]) → BaseMetric	Parse metric.
<code>parse_metric</code> (metric: typing.Union[str, typing.Type[BaseMetric], BaseMetric], task: str) → BaseMetric	Parse input metric in any form into a BaseMetric instance.
<code>parse_optimizer</code> (identifier: typing.Union[str, typing.Type[optim.Optimizer]]) → optim.Optimizer	Parse input metric in any form into a Optimizer class.

---

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]*) → 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*) → 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], Matrix: typing.Type[BaseMetric]*) → `BaseMetric`

Parse metric.

**Parameters**

- **metric** – Input metric in any form.
- **Matrix** – 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

### Classes

---

*Timer*

Computes elapsed time.

---

**class** `matchzoo.utils.timer.Timer`

Bases: `object`

Computes elapsed time.

**reset** (*self*)  
Reset timer.

**resume** (*self*)  
Resume.

**stop** (*self*)  
Stop.

**property time** (*self*)

Return time.

## Package Contents

### Classes

<i>AverageMeter</i>	Computes and stores the average and current value.
<i>Timer</i>	Computes elapsed time.
<i>EarlyStopping</i>	EarlyStopping stops training if no improvement after a given patience.

### Functions

<i>one_hot</i> (indices: int, num_classes: int) → np.ndarray	<b>return</b> A one-hot encoded vector.
<i>list_recursive_concrete_subclasses</i> (base)	List all concrete subclasses of <i>base</i> recursively.
<i>parse_loss</i> (identifier: typing.Union[str, typing.Type[nn.Module], nn.Module], task: typing.Optional[str] = None) → nn.Module	Retrieves a torch Module instance.
<i>parse_activation</i> (identifier: typing.Union[str, typing.Type[nn.Module], nn.Module]) → nn.Module	Retrieves a torch Module instance.
<i>parse_metric</i> (metric: typing.Union[str, typing.Type[BaseMetric], BaseMetric], task: str) → BaseMetric	Parse input metric in any form into a <code>BaseMetric</code> instance.
<i>parse_optimizer</i> (identifier: typing.Union[str, typing.Type[optm.Optimizer]]) → optm.Optimizer	Parse input metric in any form into a <code>Optimizer</code> class.
<i>get_file</i> (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.
<i>_hash_file</i> (fpath, algorithm='sha256', chunk_size=65535)	Calculates a file sha256 or md5 hash.

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, nn.Module]`, *task*: `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(optim.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
```

**reset** (*self*)  
Reset AverageMeter.

**update** (*self*, *val*, *n=1*)  
Update value.

**property avg** (*self*)  
Get avg.

**class** matchzoo.utils.**Timer**

Bases: object

Computes elapsed time.

**reset** (*self*)  
Reset timer.

**resume** (*self*)  
Resume.

**stop** (*self*)  
Stop.

**property time** (*self*)  
Return time.

**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 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.

**state\_dict** (*self*) → typing.Dict[str, typing.Any]  
A *Trainer* can use this to serialize the state.

**load\_state\_dict** (*self*, *state\_dict: typing.Dict[str, typing.Any]*) → None  
Hydrate a early stopping from a serialized state.

**update** (*self*, *result: list*)  
Call function.

**property best\_so\_far** (*self*) → bool  
Returns best so far.

**property is\_best\_so\_far** (*self*) → bool  
Returns true if it is the best so far.



**property should\_stop\_early** (*self*) → bool

Returns true if improvement has stopped for long enough.

`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`, *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._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__ = 1.1.1
```

### 3.1.3 Package Contents

#### Classes

<code>DataPack</code>	Matchzoo <code>DataPack</code> data structure, store dataframe and context.
<code>Param</code>	Parameter class.
<code>ParamTable</code>	Parameter table class.
<code>Embedding</code>	Embedding class.

#### Functions

<code>load_data_pack(dirpath: typing.Union[str, Path]) → DataPack</code>	Load a <code>DataPack</code> . The reverse function of <code>save()</code> .
<code>chain_transform(units: typing.List[Unit]) → typing.Callable</code>	Compose unit transformations into a single function.
<code>load_preprocessor(dirpath: typing.Union[str, Path]) → 'mz.DataPack'</code>	Load the fitted <code>context</code> . The reverse function of <code>save()</code> .
<code>build_unit_from_data_pack(unit: StatefulUnit, data_pack: mz.DataPack, mode: str = 'both', flatten: bool = True, verbose: int = 1) → StatefulUnit</code>	Build a <code>StatefulUnit</code> from a <code>DataPack</code> object.
<code>build_vocab_unit(data_pack: DataPack, mode: str = 'both', verbose: int = 1) → Vocabulary</code>	Build a <code>preprocessor.units.Vocabulary</code> given <code>data_pack</code> .

```
matchzoo.USER_DIR
```

```
matchzoo.USER_DATA_DIR
```

```
matchzoo.USER_TUNED_MODELS_DIR
```

```
matchzoo.__version__ = 1.1.1
```

```
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 = [
...     ['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]`) → `pd.DataFrame`  
Slicer.

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

`DATA_FILENAME` = `data.dill`

**property** `has_label` (`self`) → `bool`

**Returns** `True` if `label` column exists, `False` otherwise.

`__len__` (`self`) → `int`  
Get numer of rows in the class:`DataPack` object.

**property** `frame` (`self`) → `'DataPack.FrameView'`

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

```

**unpack** (*self*) → typing.Tuple[typing.Dict[str, np.array], typing.Optional[np.array]]  
 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]) → 'DataPack'  
 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*.

**property** *relation* (*self*)  
*relation* getter.

**property** *left* (*self*) → pd.DataFrame  
 Get *left()* of *DataPack*.

**property** *right* (*self*) → pd.DataFrame  
 Get *right()* of *DataPack*.

**copy** (*self*) → 'DataPack'

**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.

**drop\_empty** (*self*)

Process empty data by removing corresponding rows.

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

**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]*) → *typing.Callable*

Compose unit transformations into a single function.

**Parameters** `units` – List of `matchzoo.StatelessUnit`.

`matchzoo.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.

```
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
```

(continues on next page)

(continued from previous page)

```

... )
>>> param.validator
<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
<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'>

```

**property** `name` (`self`) → str



**Returns** Name of the parameter.

**property value** (*self*) → typing.Any

**Returns** Value of the parameter.

**property hyper\_space** (*self*) → SpaceType

**Returns** Hyper space of the parameter.

**property validator** (*self*) → typing.Callable[[typing.Any], bool]

**Returns** Validator of the parameter.

**property desc** (*self*) → str

**Returns** Parameter description.

**\_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.

```

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

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

**get** (*self*, *key*) → Param

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

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

Set *key* to parameter *param*.

**property hyper\_space** (*self*) → dict

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

**to\_frame** (*self*) → pd.DataFrame

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) → typing.Any

**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*) → typing.Iterator

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

**completed** (*self*, *exclude*: typing.Optional[list] = None) → bool

Check if all params are filled.

**Parameters** **exclude** – List of names of parameters that was excluded from being computed.

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

### Example

```
>>> import matchzoo
>>> model = matchzoo.models.DenseBaseline()
>>> model.params.completed(
...     exclude=['task', 'out_activation_func', 'embedding',
...             'embedding_input_dim', 'embedding_output_dim']
... )
True
```

**keys** (*self*) → collections.abc.KeysView

**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])  
 → np.ndarray  
 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.



## INDICES AND TABLES

- genindex
- modindex
- search





## PYTHON MODULE INDEX

### m

- matchzoo, 33
- matchzoo.auto, 33
- matchzoo.auto.preparer, 33
- matchzoo.auto.preparer.prepare, 33
- matchzoo.auto.preparer.preparer, 34
- matchzoo.auto.tuner, 37
- matchzoo.auto.tuner.tune, 37
- matchzoo.auto.tuner.tuner, 39
- matchzoo.data\_pack, 47
- matchzoo.data\_pack.data\_pack, 47
- matchzoo.data\_pack.pack, 52
- matchzoo.dataloader, 59
- matchzoo.dataloader.callbacks, 59
- matchzoo.dataloader.callbacks.histogram, 59
- matchzoo.dataloader.callbacks.lambda\_callback, 60
- matchzoo.dataloader.callbacks.ngram, 61
- matchzoo.dataloader.callbacks.padding, 62
- matchzoo.dataloader.dataloader, 67
- matchzoo.dataloader.dataloader\_builder, 68
- matchzoo.dataloader.dataset, 69
- matchzoo.dataloader.dataset\_builder, 71
- matchzoo.datasets, 75
- matchzoo.datasets.embeddings, 75
- matchzoo.datasets.embeddings.load\_fasttext\_embedding, 75
- matchzoo.datasets.embeddings.load\_glove\_embedding, 76
- matchzoo.datasets.quora\_qp, 77
- matchzoo.datasets.quora\_qp.load\_data, 77
- matchzoo.datasets.snli, 78
- matchzoo.datasets.snli.load\_data, 78
- matchzoo.datasets.toy, 80
- matchzoo.datasets.wiki\_qa, 81
- matchzoo.datasets.wiki\_qa.load\_data, 81
- matchzoo.embedding, 83
- matchzoo.embedding.embedding, 83
- matchzoo.engine, 85
- matchzoo.engine.base\_callback, 85
- matchzoo.engine.base\_metric, 86
- matchzoo.engine.base\_model, 87
- matchzoo.engine.base\_preprocessor, 90
- matchzoo.engine.base\_task, 91
- matchzoo.engine.hyper\_spaces, 92
- matchzoo.engine.param, 94
- matchzoo.engine.param\_table, 97
- matchzoo.losses, 99
- matchzoo.losses.rank\_cross\_entropy\_loss, 99
- matchzoo.losses.rank\_hinge\_loss, 100
- matchzoo.metrics, 102
- matchzoo.metrics.accuracy, 102
- matchzoo.metrics.average\_precision, 103
- matchzoo.metrics.cross\_entropy, 103
- matchzoo.metrics.discounted\_cumulative\_gain, 104
- matchzoo.metrics.mean\_average\_precision, 105
- matchzoo.metrics.mean\_reciprocal\_rank, 106
- matchzoo.metrics.normalized\_discounted\_cumulative\_gain, 107
- matchzoo.metrics.precision, 108
- matchzoo.models, 114
- matchzoo.models.anmm, 114
- matchzoo.models.arci, 114
- matchzoo.models.arcii, 116
- matchzoo.models.bert, 117
- matchzoo.models.bimpm, 118
- matchzoo.models.cdssm, 119
- matchzoo.models.conv\_knrm, 121
- matchzoo.models.dense\_baseline, 122
- matchzoo.models.diin, 122
- matchzoo.models.drmm, 124
- matchzoo.models.drmm\_tks, 124
- matchzoo.models.dssm, 126
- matchzoo.models.duet, 127
- matchzoo.models.esim, 128
- matchzoo.models.hbmp, 129

- matchzoo.models.knrm, 129
- matchzoo.models.match\_pyramid, 130
- matchzoo.models.match\_srnn, 131
- matchzoo.models.match\_lstm, 132
- matchzoo.models.mvlstm, 132
- matchzoo.models.parameter\_readme\_generator, 134
- matchzoo.modules, 148
  - attention, 148
  - bert\_module, 149
  - character\_embedding, 150
  - dense\_net, 151
  - dropout, 152
  - gaussian\_kernel, 152
  - matching, 153
  - matching\_tensor, 153
  - semantic\_composite, 154
  - spatial\_gru, 155
  - stacked\_brnn, 157
- matchzoo.preprocessors, 164
  - basic\_preprocessor, 183
  - bert\_preprocessor, 185
  - build\_unit\_from\_data\_pack, 186
  - build\_vocab\_unit, 187
  - chain\_transform, 187
  - naive\_preprocessor, 188
  - units, 164
    - character\_index, 164
    - digit\_removal, 165
    - frequency\_filter, 165
    - lemmatization, 166
    - lowercase, 167
    - matching\_histogram, 167
    - ngram\_letter, 168
    - punc\_removal, 169
    - stateful\_unit, 169
    - stemming, 170
    - stop\_removal, 170
    - tokenize, 171
    - truncated\_length, 171
    - unit, 172
    - vocabulary, 172
    - word\_exact\_match, 173
    - word\_hashing, 174
- matchzoo.tasks, 191
  - classification, 191
  - ranking, 192
- matchzoo.trainers, 195
  - trainer, 195
- matchzoo.utils, 200
  - average\_meter, 200
  - early\_stopping, 201
  - get\_file, 202
  - list\_recursive\_subclasses, 204
  - one\_hot, 205
  - parse, 205
  - tensor\_type, 208
  - timer, 208
  - version, 214

## Symbols

- `__MATCH_PUNC` (*matchzoo.preprocessors.units.PuncRemoval* attribute), 178
- `__MATCH_PUNC` (*matchzoo.preprocessors.units.punc\_removal.PuncRemoval* attribute), 169
- `__add__` () (*matchzoo.engine.hyper\_spaces.HyperoptProxy* method), 93
- `__bool__` () (*matchzoo.Param* method), 221
- `__bool__` () (*matchzoo.engine.param.Param* method), 96
- `__call__` () (*matchzoo.DataPack.FrameView* method), 215
- `__call__` () (*matchzoo.data\_pack.DataPack.FrameView* method), 54
- `__call__` () (*matchzoo.data\_pack.data\_pack.DataPack.FrameView* method), 48
- `__call__` () (*matchzoo.engine.base\_metric.BaseMetric* method), 86
- `__call__` () (*matchzoo.metrics.Accuracy* method), 112
- `__call__` () (*matchzoo.metrics.CrossEntropy* method), 113
- `__call__` () (*matchzoo.metrics.DiscountedCumulativeGain* method), 110
- `__call__` () (*matchzoo.metrics.MeanAveragePrecision* method), 111
- `__call__` () (*matchzoo.metrics.MeanReciprocalRank* method), 111
- `__call__` () (*matchzoo.metrics.NormalizedDiscountedCumulativeGain* method), 112
- `__call__` () (*matchzoo.metrics.Precision* method), 109
- `__call__` () (*matchzoo.metrics.accuracy.Accuracy* method), 102
- `__call__` () (*matchzoo.metrics.average\_precision.AveragePrecision* method), 103
- `__call__` () (*matchzoo.metrics.cross\_entropy.CrossEntropy* method), 104
- `__call__` () (*matchzoo.metrics.discounted\_cumulative\_gain.DiscountedCumulativeGain* method), 105
- `__call__` () (*matchzoo.metrics.mean\_average\_precision.MeanAveragePrecision* method), 106
- `__call__` () (*matchzoo.metrics.mean\_reciprocal\_rank.MeanReciprocalRank* method), 107
- `__call__` () (*matchzoo.metrics.normalized\_discounted\_cumulative\_gain.NormalizedDiscountedCumulativeGain* method), 107
- `__call__` () (*matchzoo.metrics.precision.Precision* method), 108
- `__constants__` (*matchzoo.losses.RankCrossEntropyLoss* attribute), 101
- `__constants__` (*matchzoo.losses.RankHingeLoss* attribute), 101
- `__constants__` (*matchzoo.losses.rank\_cross\_entropy\_loss.RankCrossEntropyLoss* attribute), 100
- `__constants__` (*matchzoo.losses.rank\_hinge\_loss.RankHingeLoss* attribute), 100
- `__contains__` () (*matchzoo.ParamTable* method), 223
- `__contains__` () (*matchzoo.engine.param\_table.ParamTable* method), 99
- `__eq__` () (*matchzoo.engine.base\_metric.BaseMetric* method), 87
- `__floordiv__` () (*matchzoo.engine.hyper\_spaces.HyperoptProxy* method), 93
- `__getitem__` () (*matchzoo.DataPack* method), 216

`__getitem__()` (*matchzoo.DataPack.FrameView method*), 215  
`__getitem__()` (*matchzoo.ParamTable method*), 222  
`__getitem__()` (*matchzoo.data\_pack.DataPack method*), 55  
`__getitem__()` (*matchzoo.data\_pack.DataPack.FrameView method*), 54  
`__getitem__()` (*matchzoo.data\_pack.data\_pack.DataPack method*), 49  
`__getitem__()` (*matchzoo.data\_pack.data\_pack.DataPack.FrameView method*), 48  
`__getitem__()` (*matchzoo.dataloader.Dataset method*), 72  
`__getitem__()` (*matchzoo.dataloader.dataset.Dataset method*), 70  
`__getitem__()` (*matchzoo.engine.param\_table.ParamTable method*), 98  
`__hash__()` (*matchzoo.engine.base\_metric.BaseMetric method*), 87  
`__iter__()` (*matchzoo.ParamTable method*), 223  
`__iter__()` (*matchzoo.dataloader.DataLoader method*), 74  
`__iter__()` (*matchzoo.dataloader.Dataset method*), 72  
`__iter__()` (*matchzoo.dataloader.dataloader.DataLoader method*), 68  
`__iter__()` (*matchzoo.dataloader.dataset.Dataset method*), 70  
`__iter__()` (*matchzoo.engine.param\_table.ParamTable method*), 98  
`__len__()` (*matchzoo.DataPack method*), 215  
`__len__()` (*matchzoo.data\_pack.DataPack method*), 54  
`__len__()` (*matchzoo.data\_pack.data\_pack.DataPack method*), 48  
`__len__()` (*matchzoo.dataloader.DataLoader method*), 74  
`__len__()` (*matchzoo.dataloader.Dataset method*), 72  
`__len__()` (*matchzoo.dataloader.dataloader.DataLoader method*), 68  
`__len__()` (*matchzoo.dataloader.dataset.Dataset method*), 70  
`__missing__()` (*matchzoo.preprocessors.units.Vocabulary.TermIndex method*), 180  
`__missing__()` (*matchzoo.preprocessors.units.vocabulary.Vocabulary.TermIndex method*), 173  
`__mul__()` (*matchzoo.engine.hyper\_spaces.HyperoptProxy method*), 93  
`__neg__()` (*matchzoo.engine.hyper\_spaces.HyperoptProxy method*), 93  
`__pow__()` (*matchzoo.engine.hyper\_spaces.HyperoptProxy method*), 93  
`__radd__()` (*matchzoo.engine.hyper\_spaces.HyperoptProxy method*), 93  
`__repr__()` (*matchzoo.engine.base\_metric.BaseMetric method*), 86  
`__repr__()` (*matchzoo.metrics.Accuracy method*), 112  
`__repr__()` (*matchzoo.metrics.CrossEntropy method*), 113  
`__repr__()` (*matchzoo.metrics.DiscountedCumulativeGain method*), 110  
`__repr__()` (*matchzoo.metrics.MeanAveragePrecision method*), 111  
`__repr__()` (*matchzoo.metrics.MeanReciprocalRank method*), 111  
`__repr__()` (*matchzoo.metrics.NormalizedDiscountedCumulativeGain method*), 112  
`__repr__()` (*matchzoo.metrics.Precision method*), 109  
`__repr__()` (*matchzoo.metrics.accuracy.Accuracy method*), 102  
`__repr__()` (*matchzoo.metrics.average\_precision.AveragePrecision method*), 103  
`__repr__()` (*matchzoo.metrics.cross\_entropy.CrossEntropy method*), 104  
`__repr__()` (*matchzoo.metrics.discounted\_cumulative\_gain.DiscountedCumulativeGain method*), 105  
`__repr__()` (*matchzoo.metrics.mean\_average\_precision.MeanAveragePrecision method*), 106  
`__repr__()` (*matchzoo.metrics.mean\_reciprocal\_rank.MeanReciprocalRank method*), 107  
`__repr__()` (*matchzoo.metrics.normalized\_discounted\_cumulative\_gain.NormalizedDiscountedCumulativeGain method*), 107  
`__repr__()` (*matchzoo.metrics.precision.Precision method*), 108  
`__rfloordiv__()` (*matchzoo.engine.hyper\_spaces.HyperoptProxy method*), 93

- `zoo.engine.hyper_spaces.HyperoptProxy` method), 93
- `__rmul__()` (`matchzoo.engine.hyper_spaces.HyperoptProxy` method), 93
- `__rpow__()` (`matchzoo.engine.hyper_spaces.HyperoptProxy` method), 93
- `__rsub__()` (`matchzoo.engine.hyper_spaces.HyperoptProxy` method), 93
- `__rtruediv__()` (`matchzoo.engine.hyper_spaces.HyperoptProxy` method), 93
- `__getitem__()` (`matchzoo.ParamTable` method), 222
- `__setitem__()` (`matchzoo.engine.param_table.ParamTable` method), 98
- `__str__()` (`matchzoo.ParamTable` method), 222
- `__str__()` (`matchzoo.engine.hyper_spaces.choice` method), 94
- `__str__()` (`matchzoo.engine.hyper_spaces.quniform` method), 94
- `__str__()` (`matchzoo.engine.hyper_spaces.uniform` method), 94
- `__str__()` (`matchzoo.engine.param_table.ParamTable` method), 98
- `__str__()` (`matchzoo.tasks.Classification` method), 194
- `__str__()` (`matchzoo.tasks.Ranking` method), 194
- `__str__()` (`matchzoo.tasks.classification.Classification` method), 192
- `__str__()` (`matchzoo.tasks.ranking.Ranking` method), 193
- `__sub__()` (`matchzoo.engine.hyper_spaces.HyperoptProxy` method), 93
- `__truediv__()` (`matchzoo.engine.hyper_spaces.HyperoptProxy` method), 93
- `__version__` (in module `matchzoo`), 214
- `__version__` (in module `matchzoo.version`), 214
- `_apply_on_text_both()` (`matchzoo.DataPack` method), 219
- `_apply_on_text_both()` (`matchzoo.data_pack.DataPack` method), 58
- `_apply_on_text_both()` (`matchzoo.data_pack.data_pack.DataPack` method), 52
- `_apply_on_text_left()` (`matchzoo.DataPack` method), 219
- `_apply_on_text_left()` (`matchzoo.data_pack.DataPack` method), 58
- `_apply_on_text_left()` (`matchzoo.data_pack.data_pack.DataPack` method), 52
- `_apply_on_text_right()` (`matchzoo.DataPack` method), 219
- `_apply_on_text_right()` (`matchzoo.data_pack.DataPack` method), 58
- `_apply_on_text_right()` (`matchzoo.data_pack.data_pack.DataPack` method), 52
- `_assure_losses()` (`matchzoo.datasets.toy.BaseTask` method), 80
- `_assure_losses()` (`matchzoo.engine.base_task.BaseTask` method), 91
- `_assure_metrics()` (`matchzoo.datasets.toy.BaseTask` method), 80
- `_assure_metrics()` (`matchzoo.engine.base_task.BaseTask` method), 91
- `_backward()` (`matchzoo.trainers.Trainer` method), 199
- `_backward()` (`matchzoo.trainers.trainer.Trainer` method), 196
- `_bfs()` (in module `matchzoo.utils.list_recursive_subclasses`), 204
- `_build_data_loader_builder()` (`matchzoo.auto.Preparer` method), 45
- `_build_data_loader_builder()` (`matchzoo.auto.preparer.Preparer` method), 37
- `_build_data_loader_builder()` (`matchzoo.auto.preparer.preparer.Preparer` method), 35
- `_build_dataset_builder()` (`matchzoo.auto.Preparer` method), 45
- `_build_dataset_builder()` (`matchzoo.auto.preparer.Preparer` method), 37
- `_build_dataset_builder()` (`matchzoo.auto.preparer.preparer.Preparer` method), 35
- `_build_match_histogram()` (in module `matchzoo.data_loader.callbacks.histogram`), 59
- `_build_matrix()` (`matchzoo.auto.Preparer` method), 45
- `_build_matrix()` (`matchzoo.auto.preparer.Preparer` method), 37
- `_build_matrix()` (`matchzoo.auto.preparer.preparer.Preparer` method), 35
- `_build_model()` (`matchzoo.auto.Preparer` method), 45
- `_build_model()` (`matchzoo.auto.preparer.Preparer` method), 37
- `_build_model()` (`matchzoo.auto.preparer.preparer.Preparer` method), 35

- `_build_word_ngram_map()` (in module `matchzoo.dataloader.callbacks.ngram`), 61
- `_convert()` (`matchzoo.datasets.toy.BaseTask` method), 80
- `_convert()` (`matchzoo.engine.base_task.BaseTask` method), 91
- `_convert_to_list_index()` (in module `matchzoo.data_pack.data_pack`), 47
- `_create_base_network()` (`matchzoo.models.CDSSM` method), 137
- `_create_base_network()` (`matchzoo.models.cdssm.CDSSM` method), 120
- `_create_full_params()` (`matchzoo.auto.Tuner` method), 46
- `_create_full_params()` (`matchzoo.auto.tuner.Tuner` method), 42
- `_create_full_params()` (`matchzoo.auto.tuner.tuner.Tuner` method), 40
- `_default_units()` (`matchzoo.engine.base_preprocessor.BasePreprocessor` class method), 91
- `_df()` (`matchzoo.preprocessors.units.FrequencyFilter` class method), 177
- `_df()` (`matchzoo.preprocessors.units.frequency_filter.FrequencyFilter` class method), 166
- `_download_data()` (in module `matchzoo.datasets.quora_qp.load_data`), 77
- `_download_data()` (in module `matchzoo.datasets.snli.load_data`), 79
- `_download_data()` (in module `matchzoo.datasets.wiki_qa.load_data`), 82
- `_eval_metric_on_data_frame()` (`matchzoo.trainers.Trainer` class method), 199
- `_eval_metric_on_data_frame()` (`matchzoo.trainers.trainer.Trainer` class method), 197
- `_extract_archive()` (in module `matchzoo.utils.get_file`), 203
- `_fasttext_embedding_url` (in module `matchzoo.datasets.embeddings.load_fasttext_embedding`), 75
- `_filter_concrete()` (in module `matchzoo.utils.list_recursive_subclasses`), 204
- `_fix_loss_sign()` (`matchzoo.auto.Tuner` method), 46
- `_fix_loss_sign()` (`matchzoo.auto.tuner.Tuner` method), 42
- `_fix_loss_sign()` (`matchzoo.auto.tuner.tuner.Tuner` method), 40
- `_fmin()` (`matchzoo.auto.Tuner` method), 46
- `_fmin()` (`matchzoo.auto.tuner.Tuner` method), 42
- `_fmin()` (`matchzoo.auto.tuner.tuner.Tuner` method), 40
- `_forward_unpadded()` (`matchzoo.modules.StackedBRNN` method), 160
- `_forward_unpadded()` (`matchzoo.modules.stacked_brnn.StackedBRNN` method), 157
- `_gen_ids()` (in module `matchzoo.data_pack.pack`), 53
- `_generate()` (in module `matchzoo.models.parameter_readme_generator`), 134
- `_glove_embedding_url` (in module `matchzoo.datasets.embeddings.load_glove_embedding`), 76
- `_handle_callbacks_on_batch_data_pack()` (`matchzoo.dataloader.Dataset` method), 72
- `_handle_callbacks_on_batch_data_pack()` (`matchzoo.dataloader.dataset.Dataset` method), 70
- `_handle_callbacks_on_batch_unpacked()` (`matchzoo.dataloader.DataLoader` method), 74
- `_handle_callbacks_on_batch_unpacked()` (`matchzoo.dataloader.Dataset` method), 72
- `_handle_callbacks_on_batch_unpacked()` (`matchzoo.dataloader.dataloader.DataLoader` method), 68
- `_handle_callbacks_on_batch_unpacked()` (`matchzoo.dataloader.dataset.Dataset` method), 70
- `_hash_file()` (in module `matchzoo.utils`), 213
- `_hash_file()` (in module `matchzoo.utils.get_file`), 204
- `_idf()` (`matchzoo.preprocessors.units.FrequencyFilter` class method), 177
- `_idf()` (`matchzoo.preprocessors.units.frequency_filter.FrequencyFilter` class method), 166
- `_infer_dtype()` (in module `matchzoo.dataloader.callbacks.padding`), 62
- `_infer_num_neg()` (`matchzoo.auto.Preparer` method), 45
- `_infer_num_neg()` (`matchzoo.auto.preparer.Preparer` method), 37
- `_infer_num_neg()` (`matchzoo.auto.preparer.preparer.Preparer` method), 35
- `_infer_pre_assignment_hook()` (`matchzoo.Param` method), 221
- `_infer_pre_assignment_hook()` (`matchzoo.engine.param.Param` method), 96
- `_load_dataloader()` (`matchzoo.trainers.Trainer` method), 198
- `_load_dataloader()` (`matchzoo.trainers.trainer.Trainer` method), 196
- `_load_model()` (`matchzoo.trainers.Trainer` method), 199
- `_load_model()` (`matchzoo.trainers.trainer.Trainer` method), 196
- `_load_path()` (`matchzoo.trainers.Trainer` method),

- 199
- `_load_path()` (*matchzoo.trainers.trainer.Trainer* method), 196
- `_log_result()` (*matchzoo.auto.Tuner* class method), 46
- `_log_result()` (*matchzoo.auto.tuner.Tuner* class method), 42
- `_log_result()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 40
- `_make_conv_block()` (*matchzoo.modules.dense\_net.DenseBlock* class method), 151
- `_make_conv_pool_block()` (*matchzoo.models.ArcI* class method), 141
- `_make_conv_pool_block()` (*matchzoo.models.ArcII* class method), 142
- `_make_conv_pool_block()` (*matchzoo.models.MatchPyramid* class method), 144
- `_make_conv_pool_block()` (*matchzoo.models.arci.ArcI* class method), 115
- `_make_conv_pool_block()` (*matchzoo.models.arcii.ArcII* class method), 117
- `_make_conv_pool_block()` (*matchzoo.models.match\_pyramid.MatchPyramid* class method), 131
- `_make_default_embedding_layer()` (*matchzoo.engine.base\_model.BaseModel* method), 89
- `_make_doc_section_subsubtitle()` (in module *matchzoo.models.parameter\_readme\_generator*), 134
- `_make_embedding_layer()` (*matchzoo.engine.base\_model.BaseModel* method), 89
- `_make_model_class_subtitle()` (in module *matchzoo.models.parameter\_readme\_generator*), 134
- `_make_model_doc()` (in module *matchzoo.models.parameter\_readme\_generator*), 134
- `_make_model_params_table()` (in module *matchzoo.models.parameter\_readme\_generator*), 134
- `_make_multi_layer_perceptron_layer()` (*matchzoo.engine.base\_model.BaseModel* method), 89
- `_make_output_layer()` (*matchzoo.engine.base\_model.BaseModel* method), 89
- `_make_params_section_subsubtitle()` (in module *matchzoo.models.parameter\_readme\_generator*), 134
- `_make_perceptron_layer()` (*matchzoo.engine.base\_model.BaseModel* method), 89
- `_make_title()` (in module *matchzoo.models.parameter\_readme\_generator*), 134
- `_make_transition_block()` (*matchzoo.modules.DenseNet* class method), 162
- `_make_transition_block()` (*matchzoo.modules.dense\_net.DenseNet* class method), 151
- `_merge()` (in module *matchzoo.data\_pack.pack*), 53
- `_normalize_embedding()` (*matchzoo.preprocessors.units.MatchingHistogram* method), 178
- `_normalize_embedding()` (*matchzoo.preprocessors.units.matching\_histogram.MatchingHistogram* method), 168
- `_optional_inplace()` (*matchzoo.DataPack* method), 217
- `_optional_inplace()` (*matchzoo.data\_pack.DataPack* method), 56
- `_optional_inplace()` (*matchzoo.data\_pack.data\_pack.DataPack* method), 50
- `_padding_2D()` (in module *matchzoo.dataloader.callbacks.padding*), 62
- `_padding_3D()` (in module *matchzoo.dataloader.callbacks.padding*), 62
- `_parse()` (in module *matchzoo.utils.parse*), 205
- `_parse_metric()` (in module *matchzoo.utils.parse*), 207
- `_read_data()` (in module *matchzoo.datasets.quora\_qp.load\_data*), 77
- `_read_data()` (in module *matchzoo.datasets.snli.load\_data*), 79
- `_read_data()` (in module *matchzoo.datasets.wiki\_qa.load\_data*), 82
- `_reorganize_pair_wise()` (*matchzoo.dataloader.Dataset* class method), 73
- `_reorganize_pair_wise()` (*matchzoo.dataloader.dataset.Dataset* class method), 70
- `_run()` (*matchzoo.auto.Tuner* method), 46
- `_run()` (*matchzoo.auto.tuner.Tuner* method), 42
- `_run()` (*matchzoo.auto.tuner.tuner.Tuner* method), 40
- `_run_epoch()` (*matchzoo.trainers.Trainer* method), 199
- `_run_epoch()` (*matchzoo.trainers.trainer.Trainer* method), 196
- `_run_scheduler()` (*matchzoo.trainers.Trainer* method), 199

- `_run_scheduler()` (*matchzoo.trainers.trainer.Trainer* method), 196
  - `_save()` (*matchzoo.trainers.Trainer* method), 200
  - `_save()` (*matchzoo.trainers.trainer.Trainer* method), 197
  - `_set_param_default()` (*matchzoo.engine.base\_model.BaseModel* method), 88
  - `_tf()` (*matchzoo.preprocessors.units.FrequencyFilter* class method), 177
  - `_tf()` (*matchzoo.preprocessors.units.frequency\_filter.FrequencyFilter* class method), 166
  - `_trunc_text()` (in module *matchzoo.dataloader.callbacks.histogram*), 59
  - `_url` (in module *matchzoo.datasets.quora\_qp.load\_data*), 77
  - `_url` (in module *matchzoo.datasets.snli.load\_data*), 78
  - `_url` (in module *matchzoo.datasets.wiki\_qa.load\_data*), 81
  - `_validate()` (*matchzoo.Param* method), 221
  - `_validate()` (*matchzoo.engine.param.Param* method), 96
  - `_validate_dataloader()` (*matchzoo.auto.Tuner* class method), 46
  - `_validate_dataloader()` (*matchzoo.auto.tuner.Tuner* class method), 42
  - `_validate_dataloader()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 41
  - `_validate_kwargs()` (*matchzoo.auto.Tuner* class method), 47
  - `_validate_kwargs()` (*matchzoo.auto.tuner.Tuner* class method), 42
  - `_validate_kwargs()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 41
  - `_validate_matching_type()` (*matchzoo.modules.Matching* class method), 160
  - `_validate_matching_type()` (*matchzoo.modules.matching.Matching* class method), 153
  - `_validate_metric()` (*matchzoo.auto.Tuner* class method), 47
  - `_validate_metric()` (*matchzoo.auto.tuner.Tuner* class method), 42
  - `_validate_metric()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 41
  - `_validate_mode()` (*matchzoo.auto.Tuner* class method), 47
  - `_validate_mode()` (*matchzoo.auto.tuner.Tuner* class method), 42
  - `_validate_mode()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 41
  - `_validate_num_runs()` (*matchzoo.auto.Tuner* class method), 47
  - `_validate_num_runs()` (*matchzoo.auto.tuner.Tuner* class method), 42
  - `_validate_num_runs()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 41
  - `_validate_optimizer()` (*matchzoo.auto.Tuner* class method), 46
  - `_validate_optimizer()` (*matchzoo.auto.tuner.Tuner* class method), 42
  - `_validate_optimizer()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 41
  - `_validate_params()` (*matchzoo.auto.Tuner* class method), 46
  - `_validate_params()` (*matchzoo.auto.tuner.Tuner* class method), 42
  - `_validate_params()` (*matchzoo.auto.tuner.tuner.Tuner* class method), 40
  - `_wrap_as_composite_func()` (in module *matchzoo.engine.hyper\_spaces*), 93
  - `_write_to_files()` (in module *matchzoo.models.parameter\_readme\_generator*), 134
  - `_xor_match()` (*matchzoo.models.DUET* class method), 146
  - `_xor_match()` (*matchzoo.models.duet.DUET* class method), 128
- ## A
- `Accuracy` (class in *matchzoo.metrics*), 112
  - `Accuracy` (class in *matchzoo.metrics.accuracy*), 102
  - `activation` (in module *matchzoo.utils.parse*), 205
  - `add()` (*matchzoo.engine.param\_table.ParamTable* method), 97
  - `add()` (*matchzoo.ParamTable* method), 222
  - `ALIAS` (*matchzoo.engine.base\_metric.BaseMetric* attribute), 86
  - `ALIAS` (*matchzoo.engine.base\_metric.ClassificationMetric* attribute), 87
  - `ALIAS` (*matchzoo.engine.base\_metric.RankingMetric* attribute), 87
  - `ALIAS` (*matchzoo.metrics.Accuracy* attribute), 112
  - `ALIAS` (*matchzoo.metrics.accuracy.Accuracy* attribute), 102
  - `ALIAS` (*matchzoo.metrics.average\_precision.AveragePrecision* attribute), 103
  - `ALIAS` (*matchzoo.metrics.cross\_entropy.CrossEntropy* attribute), 104
  - `ALIAS` (*matchzoo.metrics.CrossEntropy* attribute), 113



- ALIAS (*matchzoo.metrics.discounted\_cumulative\_gain.DiscountedCumulativeGain* attribute), 105
- ALIAS (*matchzoo.metrics.DiscountedCumulativeGain* attribute), 110
- ALIAS (*matchzoo.metrics.mean\_average\_precision.MeanAveragePrecision* attribute), 106
- ALIAS (*matchzoo.metrics.mean\_reciprocal\_rank.MeanReciprocalRank* attribute), 106
- ALIAS (*matchzoo.metrics.MeanAveragePrecision* attribute), 111
- ALIAS (*matchzoo.metrics.MeanReciprocalRank* attribute), 111
- ALIAS (*matchzoo.metrics.normalized\_discounted\_cumulative\_gain.NormalizedDiscountedCumulativeGain* attribute), 107
- ALIAS (*matchzoo.metrics.NormalizedDiscountedCumulativeGain* attribute), 112
- ALIAS (*matchzoo.metrics.Precision* attribute), 109
- ALIAS (*matchzoo.metrics.precision.Precision* attribute), 108
- aNMM (class in *matchzoo.models*), 144
- aNMM (class in *matchzoo.models.anmm*), 114
- append\_text\_length() (*matchzoo.data\_pack.data\_pack.DataPack* method), 50
- append\_text\_length() (*matchzoo.data\_pack.DataPack* method), 56
- append\_text\_length() (*matchzoo.DataPack* method), 217
- apply\_on\_text() (*matchzoo.data\_pack.data\_pack.DataPack* method), 51
- apply\_on\_text() (*matchzoo.data\_pack.DataPack* method), 57
- apply\_on\_text() (*matchzoo.DataPack* method), 218
- ArcI (class in *matchzoo.models*), 141
- ArcI (class in *matchzoo.models.arci*), 115
- ArcII (class in *matchzoo.models*), 141
- ArcII (class in *matchzoo.models.arcii*), 116
- Attention (class in *matchzoo.modules*), 158
- Attention (class in *matchzoo.modules.attention*), 148
- attention() (in module *matchzoo.models.bimpm*), 119
- AverageMeter (class in *matchzoo.utils*), 211
- AverageMeter (class in *matchzoo.utils.average\_meter*), 201
- AveragePrecision (class in *matchzoo.metrics.average\_precision*), 103
- avg() (*matchzoo.utils.average\_meter.AverageMeter* property), 201
- avg() (*matchzoo.utils.AverageMeter* property), 212
- B**
- BaseCallback (class in *matchzoo.engine.base\_callback*), 85
- BaseModel (class in *matchzoo.engine.base\_model*), 87
- BasePreprocessor (class in *matchzoo.engine.base\_preprocessor*), 90
- BaseTask (class in *matchzoo.datasets.toy*), 80
- BasicPadding (class in *matchzoo.data\_loader.callbacks*), 65
- BasicPadding (class in *matchzoo.data\_loader.callbacks.padding*), 62
- BasicPreprocessor (class in *matchzoo.preprocessors.basic\_preprocessor*), 183
- batch\_indices() (*matchzoo.data\_loader.Dataset* property), 73
- batch\_indices() (*matchzoo.data\_loader.dataset.Dataset* property), 70
- batch\_size() (*matchzoo.data\_loader.Dataset* property), 73
- batch\_size() (*matchzoo.data\_loader.dataset.Dataset* property), 70
- Bert (class in *matchzoo.models*), 142
- Bert (class in *matchzoo.models.bert*), 117
- BertModule (class in *matchzoo.modules*), 160
- BertModule (class in *matchzoo.modules.bert\_module*), 149
- BertPadding (class in *matchzoo.data\_loader.callbacks*), 66
- BertPadding (class in *matchzoo.data\_loader.callbacks.padding*), 63
- BertPreprocessor (class in *matchzoo.preprocessors*), 191
- BertPreprocessor (class in *matchzoo.preprocessors.bert\_preprocessor*), 185
- best\_so\_far() (*matchzoo.utils.early\_stopping.EarlyStopping* property), 202
- best\_so\_far() (*matchzoo.utils.EarlyStopping* property), 212
- BidirectionalAttention (class in *matchzoo.modules*), 158
- BidirectionalAttention (class in *matchzoo.modules.attention*), 148
- BiMPM (class in *matchzoo.models*), 140
- BiMPM (class in *matchzoo.models.bimpm*), 118
- build() (*matchzoo.data\_loader.data\_loader\_builder.DataLoaderBuilder* method), 68
- build() (*matchzoo.data\_loader.DataLoaderBuilder* method), 74
- build() (*matchzoo.data\_loader.dataset\_builder.DatasetBuilder* method), 71

- `build()` (*matchzoo.dataloader.DatasetBuilder* method), 75  
`build()` (*matchzoo.engine.base\_model.BaseModel* method), 89  
`build()` (*matchzoo.models.aNMM* method), 144  
`build()` (*matchzoo.models.anmm.aNMM* method), 114  
`build()` (*matchzoo.models.ArcI* method), 141  
`build()` (*matchzoo.models.arci.ArcI* method), 115  
`build()` (*matchzoo.models.ArcII* method), 142  
`build()` (*matchzoo.models.arcii.ArcII* method), 116  
`build()` (*matchzoo.models.Bert* method), 143  
`build()` (*matchzoo.models.bert.Bert* method), 117  
`build()` (*matchzoo.models.BiMPM* method), 140  
`build()` (*matchzoo.models.bimpm.BiMPM* method), 118  
`build()` (*matchzoo.models.CDSSM* method), 137  
`build()` (*matchzoo.models.cdssm.CDSSM* method), 120  
`build()` (*matchzoo.models.conv\_knrm.ConvKNRM* method), 121  
`build()` (*matchzoo.models.ConvKNRM* method), 140  
`build()` (*matchzoo.models.dense\_baseline.DenseBaseline* method), 122  
`build()` (*matchzoo.models.DenseBaseline* method), 135  
`build()` (*matchzoo.models.DIIN* method), 147  
`build()` (*matchzoo.models.diin.DIIN* method), 123  
`build()` (*matchzoo.models.DRMM* method), 138  
`build()` (*matchzoo.models.drmm.DRMM* method), 124  
`build()` (*matchzoo.models.DRMMTKS* method), 138  
`build()` (*matchzoo.models.drmmtk.DRMMTKS* method), 125  
`build()` (*matchzoo.models.DSSM* method), 136  
`build()` (*matchzoo.models.dssm.DSSM* method), 126  
`build()` (*matchzoo.models.DUET* method), 146  
`build()` (*matchzoo.models.duet.DUET* method), 128  
`build()` (*matchzoo.models.ESIM* method), 139  
`build()` (*matchzoo.models.esim.ESIM* method), 128  
`build()` (*matchzoo.models.HBMP* method), 145  
`build()` (*matchzoo.models.hbmp.HBMP* method), 129  
`build()` (*matchzoo.models.KNRM* method), 139  
`build()` (*matchzoo.models.knrm.KNRM* method), 130  
`build()` (*matchzoo.models.match\_pyramid.MatchPyramid* method), 131  
`build()` (*matchzoo.models.match\_srn.MatchSRNN* method), 132  
`build()` (*matchzoo.models.MatchLSTM* method), 140  
`build()` (*matchzoo.models.matchlst.MatchLSTM* method), 132  
`build()` (*matchzoo.models.MatchPyramid* method), 144  
`build()` (*matchzoo.models.MatchSRNN* method), 147  
`build()` (*matchzoo.models.MVLSTM* method), 143  
`build()` (*matchzoo.models.mvlstm.MVLSTM* method), 133  
`build_matrix()` (*matchzoo.Embedding* method), 224  
`build_matrix()` (*matchzoo.embedding.Embedding* method), 85  
`build_matrix()` (*matchzoo.embedding.embedding.Embedding* method), 83  
`build_unit_from_data_pack()` (*in module matchzoo*), 224  
`build_unit_from_data_pack()` (*in module matchzoo.preprocessors.build\_unit\_from\_data\_pack*), 186  
`build_vocab_unit()` (*in module matchzoo*), 224  
`build_vocab_unit()` (*in module matchzoo.preprocessors.build\_vocab\_unit*), 187
- ## C
- `calculate_recurrent_unit()` (*matchzoo.modules.spatial\_gru.SpatialGRU* method), 156  
`calculate_recurrent_unit()` (*matchzoo.modules.SpatialGRU* method), 163  
`callbacks()` (*matchzoo.dataloader.Dataset* property), 72  
`callbacks()` (*matchzoo.dataloader.dataset.Dataset* property), 70  
CDSSM (*class in matchzoo.models*), 136  
CDSSM (*class in matchzoo.models.cdssm*), 119  
`chain_transform()` (*in module matchzoo*), 219  
`chain_transform()` (*in module matchzoo.preprocessors.chain\_transform*), 187  
CharacterEmbedding (*class in matchzoo.modules*), 161  
CharacterEmbedding (*class in matchzoo.modules.character\_embedding*), 150  
CharacterIndex (*class in matchzoo.preprocessors.units*), 181  
CharacterIndex (*class in matchzoo.preprocessors.units.character\_index*), 164  
choice (*class in matchzoo.engine.hyper\_spaces*), 93  
Classification (*class in matchzoo.tasks*), 193  
Classification (*class in matchzoo.tasks.classification*), 191  
ClassificationMetric (*class in matchzoo.engine.base\_metric*), 87  
`completed()` (*matchzoo.engine.param\_table.ParamTable* method), 98  
`completed()` (*matchzoo.ParamTable* method), 223

- context () (*matchzoo.engine.base\_preprocessor.BasePreprocessor* property), 90
- context () (*matchzoo.preprocessors.units.stateful\_unit.StatefulUnit* property), 169
- context () (*matchzoo.preprocessors.units.StatefulUnit* property), 179
- convert () (*matchzoo.engine.hyper\_spaces.HyperoptProxy* method), 93
- ConvKNRM (*class in matchzoo.models*), 139
- ConvKNRM (*class in matchzoo.models.conv\_knrm*), 121
- copy () (*matchzoo.data\_pack.data\_pack.DataPack* method), 49
- copy () (*matchzoo.data\_pack.DataPack* method), 55
- copy () (*matchzoo.DataPack* method), 216
- CrossEntropy (*class in matchzoo.metrics*), 113
- CrossEntropy (*class in matchzoo.metrics.cross\_entropy*), 104
- ## D
- DATA\_FILENAME (*matchzoo.data\_pack.data\_pack.DataPack* attribute), 48
- DATA\_FILENAME (*matchzoo.data\_pack.DataPack* attribute), 54
- DATA\_FILENAME (*matchzoo.DataPack* attribute), 215
- DATA\_FILENAME (*matchzoo.engine.base\_preprocessor.BasePreprocessor* attribute), 90
- DATA\_ROOT (*in module matchzoo.datasets.embeddings*), 77
- DataLoader (*class in matchzoo.dataloader*), 73
- DataLoader (*class in matchzoo.dataloader.dataloader*), 67
- DataLoaderBuilder (*class in matchzoo.dataloader*), 74
- DataLoaderBuilder (*class in matchzoo.dataloader.dataloader\_builder*), 68
- DataPack (*class in matchzoo*), 214
- DataPack (*class in matchzoo.data\_pack*), 53
- DataPack (*class in matchzoo.data\_pack.data\_pack*), 47
- DataPack.FrameView (*class in matchzoo*), 215
- DataPack.FrameView (*class in matchzoo.data\_pack*), 54
- DataPack.FrameView (*class in matchzoo.data\_pack.data\_pack*), 48
- Dataset (*class in matchzoo.dataloader*), 71
- Dataset (*class in matchzoo.dataloader.dataset*), 69
- DatasetBuilder (*class in matchzoo.dataloader*), 74
- DatasetBuilder (*class in matchzoo.dataloader.dataset\_builder*), 71
- DenseBaseline (*class in matchzoo.models*), 135
- DenseBaseline (*class in matchzoo.models.dense\_baseline*), 122
- DenseBlock (*class in matchzoo.modules.dense\_net*), 151
- DenseNet (*class in matchzoo.modules.dense\_net*), 162
- DenseNet (*class in matchzoo.modules.dense\_net*), 151
- desc () (*matchzoo.engine.param.Param* property), 96
- desc () (*matchzoo.Param* property), 221
- DigitRemoval (*class in matchzoo.preprocessors.units*), 176
- DigitRemoval (*class in matchzoo.preprocessors.units.digit\_removal*), 165
- DIIN (*class in matchzoo.models*), 146
- DIIN (*class in matchzoo.models.diin*), 122
- DiscountedCumulativeGain (*class in matchzoo.metrics*), 110
- DiscountedCumulativeGain (*class in matchzoo.metrics.discounted\_cumulative\_gain*), 104
- div\_with\_small\_value () (*in module matchzoo.models.bimpm*), 119
- DRMM (*class in matchzoo.models*), 137
- DRMM (*class in matchzoo.models.drmm*), 124
- DRMMPadding (*class in matchzoo.dataloader.callbacks*), 66
- DRMMPadding (*class in matchzoo.dataloader.callbacks.padding*), 63
- DRMMTKS (*class in matchzoo.models*), 138
- DRMMTKS (*class in matchzoo.models.drmmtks*), 125
- drop\_empty () (*matchzoo.data\_pack.data\_pack.DataPack* method), 50
- drop\_empty () (*matchzoo.data\_pack.DataPack* method), 56
- drop\_empty () (*matchzoo.DataPack* method), 217
- drop\_label () (*matchzoo.data\_pack.data\_pack.DataPack* method), 50
- drop\_label () (*matchzoo.data\_pack.DataPack* method), 56
- drop\_label () (*matchzoo.DataPack* method), 217
- dropout () (*matchzoo.models.BiMPM* method), 140
- dropout () (*matchzoo.models.bimpm.BiMPM* method), 118
- DSSM (*class in matchzoo.models*), 135
- DSSM (*class in matchzoo.models.dssm*), 126
- DUET (*class in matchzoo.models*), 145
- DUET (*class in matchzoo.models.duet*), 127
- ## E
- EarlyStopping (*class in matchzoo.utils*), 212
- EarlyStopping (*class in matchzoo.utils.early\_stopping*), 201
- EMBED\_10 (*in module matchzoo.datasets.embeddings*), 77

EMBED\_10\_GLOVE (in module *matchzoo.datasets.embeddings*), 77

EMBED\_RANK (in module *matchzoo.datasets.embeddings*), 77

Embedding (class in *matchzoo*), 223

Embedding (class in *matchzoo.embedding*), 84

Embedding (class in *matchzoo.embedding.embedding*), 83

ESIM (class in *matchzoo.models*), 138

ESIM (class in *matchzoo.models.esim*), 128

evaluate() (*matchzoo.trainers.Trainer* method), 199

evaluate() (*matchzoo.trainers.trainer.Trainer* method), 197

## F

fit() (*matchzoo.engine.base\_preprocessor.BasePreprocessor* method), 90

fit() (*matchzoo.preprocessors.basic\_preprocessor.BasicPreprocessor* method), 185

fit() (*matchzoo.preprocessors.BasicPreprocessor* method), 190

fit() (*matchzoo.preprocessors.bert\_preprocessor.BertPreprocessor* method), 185

fit() (*matchzoo.preprocessors.BertPreprocessor* method), 191

fit() (*matchzoo.preprocessors.naive\_preprocessor.NaivePreprocessor* method), 188

fit() (*matchzoo.preprocessors.NaivePreprocessor* method), 189

fit() (*matchzoo.preprocessors.units.frequency\_filter.FrequencyFilter* method), 166

fit() (*matchzoo.preprocessors.units.FrequencyFilter* method), 177

fit() (*matchzoo.preprocessors.units.stateful\_unit.StatefulUnit* method), 169

fit() (*matchzoo.preprocessors.units.StatefulUnit* method), 179

fit() (*matchzoo.preprocessors.units.Vocabulary* method), 180

fit() (*matchzoo.preprocessors.units.vocabulary.Vocabulary* method), 173

fit\_kwargs() (*matchzoo.auto.Tuner* property), 46

fit\_kwargs() (*matchzoo.auto.tuner.Tuner* property), 42

fit\_kwargs() (*matchzoo.auto.tuner.tuner.Tuner* property), 40

fit\_transform() (*matchzoo.engine.base\_preprocessor.BasePreprocessor* method), 90

forward() (*matchzoo.engine.base\_model.BaseModel* method), 89

forward() (*matchzoo.losses.rank\_cross\_entropy\_loss.RankCrossEntropyLoss* method), 100

forward() (*matchzoo.losses.rank\_hinge\_loss.RankHingeLoss* method), 100

forward() (*matchzoo.losses.RankCrossEntropyLoss* method), 101

forward() (*matchzoo.losses.RankHingeLoss* method), 101

forward() (*matchzoo.models.aNMM* method), 144

forward() (*matchzoo.models.anmm.aNMM* method), 114

forward() (*matchzoo.models.ArcI* method), 141

forward() (*matchzoo.models.arci.ArcI* method), 115

forward() (*matchzoo.models.ArcII* method), 142

forward() (*matchzoo.models.arcii.ArcII* method), 117

forward() (*matchzoo.models.Bert* method), 143

forward() (*matchzoo.models.bert.Bert* method), 117

forward() (*matchzoo.models.BiMPM* method), 140

forward() (*matchzoo.models.bimpm.BiMPM* method), 118

forward() (*matchzoo.models.CDSSM* method), 137

forward() (*matchzoo.models.cdssm.CDSSM* method), 120

forward() (*matchzoo.models.cdssm.Squeeze* method), 121

forward() (*matchzoo.models.conv\_knrm.ConvKNRM* method), 121

forward() (*matchzoo.models.ConvKNRM* method), 140

forward() (*matchzoo.models.dense\_baseline.DenseBaseline* method), 122

forward() (*matchzoo.models.DenseBaseline* method), 135

forward() (*matchzoo.models.DIIN* method), 147

forward() (*matchzoo.models.diin.DIIN* method), 123

forward() (*matchzoo.models.DRMM* method), 138

forward() (*matchzoo.models.drmm.DRMM* method), 124

forward() (*matchzoo.models.DRMMTKS* method), 138

forward() (*matchzoo.models.drmmtnks.DRMMTKS* method), 125

forward() (*matchzoo.models.DSSM* method), 136

forward() (*matchzoo.models.dssm.DSSM* method), 126

forward() (*matchzoo.models.DUET* method), 146

forward() (*matchzoo.models.duet.DUET* method), 128

forward() (*matchzoo.models.ESIM* method), 139

forward() (*matchzoo.models.esim.ESIM* method), 128

forward() (*matchzoo.models.HBMP* method), 145

forward() (*matchzoo.models.hbmp.HBMP* method), 129

forward() (*matchzoo.models.KNRM* method), 139

forward() (*matchzoo.models.knrm.KNRM* method), 130

- `forward()` (*matchzoo.models.match\_pyramid.MatchPyramid method*), 131
- `forward()` (*matchzoo.models.match\_srnn.MatchSRNN method*), 132
- `forward()` (*matchzoo.models.MatchLSTM method*), 141
- `forward()` (*matchzoo.models.matchlstm.MatchLSTM method*), 132
- `forward()` (*matchzoo.models.MatchPyramid method*), 144
- `forward()` (*matchzoo.models.MatchSRNN method*), 147
- `forward()` (*matchzoo.models.MVLSTM method*), 143
- `forward()` (*matchzoo.models.mvlstm.MVLSTM method*), 133
- `forward()` (*matchzoo.modules.Attention method*), 158
- `forward()` (*matchzoo.modules.attention.Attention method*), 148
- `forward()` (*matchzoo.modules.attention.BidirectionalAttention method*), 148
- `forward()` (*matchzoo.modules.attention.MatchModule method*), 149
- `forward()` (*matchzoo.modules.bert\_module.BertModule method*), 149
- `forward()` (*matchzoo.modules.BertModule method*), 161
- `forward()` (*matchzoo.modules.BidirectionalAttention method*), 158
- `forward()` (*matchzoo.modules.character\_embedding.CharacterEmbedding method*), 150
- `forward()` (*matchzoo.modules.CharacterEmbedding method*), 161
- `forward()` (*matchzoo.modules.dense\_net.DenseBlock method*), 151
- `forward()` (*matchzoo.modules.dense\_net.DenseNet method*), 151
- `forward()` (*matchzoo.modules.DenseNet method*), 162
- `forward()` (*matchzoo.modules.dropout.RNNDropout method*), 152
- `forward()` (*matchzoo.modules.gaussian\_kernel.GaussianKernel method*), 152
- `forward()` (*matchzoo.modules.GaussianKernel method*), 160
- `forward()` (*matchzoo.modules.Matching method*), 160
- `forward()` (*matchzoo.modules.matching.Matching method*), 153
- `forward()` (*matchzoo.modules.matching\_tensor.MatchingTensor method*), 154
- `forward()` (*matchzoo.modules.MatchingTensor method*), 163
- `forward()` (*matchzoo.modules.MatchModule method*), 159
- `forward()` (*matchzoo.modules.RNNDropout method*), 159
- `forward()` (*matchzoo.modules.semantic\_composite.SemanticComposite method*), 155
- `forward()` (*matchzoo.modules.SemanticComposite method*), 162
- `forward()` (*matchzoo.modules.spatial\_gru.SpatialGRU method*), 156
- `forward()` (*matchzoo.modules.SpatialGRU method*), 164
- `forward()` (*matchzoo.modules.stacked\_brnn.StackedBRNN method*), 157
- `forward()` (*matchzoo.modules.StackedBRNN method*), 159
- `frame()` (*matchzoo.data\_pack.data\_pack.DataPack property*), 48
- `frame()` (*matchzoo.data\_pack.DataPack property*), 54
- `frame()` (*matchzoo.DataPack property*), 215
- `FrequencyFilter` (class in *matchzoo.preprocessors.units*), 176
- `FrequencyFilter` (class in *matchzoo.preprocessors.units.frequency\_filter*), 165
- ## G
- `GaussianKernel` (class in *matchzoo.modules*), 160
- `GaussianKernel` (class in *matchzoo.modules.gaussian\_kernel*), 152
- `get()` (*matchzoo.engine.param\_table.ParamTable method*), 97
- `get()` (*matchzoo.engine.param\_table.ParamTable method*), 222
- `get_default_config()` (*matchzoo.auto.Preparer class method*), 45
- `get_default_config()` (*matchzoo.auto.preparer.Preparer class method*), 37
- `get_default_config()` (*matchzoo.auto.preparer.preparer.Preparer class method*), 35
- `get_default_padding_callback()` (*matchzoo.engine.base\_model.BaseModel class*)
- `get_default_padding_callback()` (*matchzoo.models.ArcI class method*), 141
- `get_default_padding_callback()` (*matchzoo.models.arci.ArcI class method*), 115
- `get_default_padding_callback()` (*matchzoo.models.ArcII class method*), 142
- `get_default_padding_callback()` (*matchzoo.models.arcii.ArcII class method*), 116
- `get_default_padding_callback()` (*matchzoo.models.Bert class method*), 142
- `get_default_padding_callback()` (*matchzoo.models.bert.Bert class method*), 117
- `get_default_padding_callback()` (*matchzoo.models.CDSSM class method*), 137

`get_default_padding_callback()` (*matchzoo.models.cdssm.CDSSM class method*), 120  
`get_default_padding_callback()` (*matchzoo.models.DIIN class method*), 147  
`get_default_padding_callback()` (*matchzoo.models.diin.DIIN class method*), 123  
`get_default_padding_callback()` (*matchzoo.models.DRMM class method*), 137  
`get_default_padding_callback()` (*matchzoo.models.drmm.DRMM class method*), 124  
`get_default_padding_callback()` (*matchzoo.models.DRMMTKS class method*), 138  
`get_default_padding_callback()` (*matchzoo.models.drmmtk.DRMMTKS class method*), 125  
`get_default_padding_callback()` (*matchzoo.models.DSSM class method*), 136  
`get_default_padding_callback()` (*matchzoo.models.dssm.DSSM class method*), 126  
`get_default_padding_callback()` (*matchzoo.models.DUET class method*), 146  
`get_default_padding_callback()` (*matchzoo.models.duet.DUET class method*), 127  
`get_default_padding_callback()` (*matchzoo.models.MVLSTM class method*), 143  
`get_default_padding_callback()` (*matchzoo.models.mvlstm.MVLSTM class method*), 133  
`get_default_params()` (*matchzoo.engine.base\_model.BaseModel class method*), 88  
`get_default_params()` (*matchzoo.models.aNMM class method*), 144  
`get_default_params()` (*matchzoo.models.anmm.aNMM class method*), 114  
`get_default_params()` (*matchzoo.models.ArcI class method*), 141  
`get_default_params()` (*matchzoo.models.arci.ArcI class method*), 115  
`get_default_params()` (*matchzoo.models.ArcII class method*), 142  
`get_default_params()` (*matchzoo.models.arcii.ArcII class method*), 116  
`get_default_params()` (*matchzoo.models.Bert class method*), 142  
`get_default_params()` (*matchzoo.models.bert.Bert class method*), 117  
`get_default_params()` (*matchzoo.models.BiMPM class method*), 140  
`get_default_params()` (*matchzoo.models.bimpm.BiMPM class method*), 118  
`get_default_params()` (*matchzoo.models.CDSSM class method*), 136  
`get_default_params()` (*matchzoo.models.cdssm.CDSSM class method*), 120  
`get_default_params()` (*matchzoo.models.conv\_knrm.ConvKNRM class method*), 121  
`get_default_params()` (*matchzoo.models.ConvKNRM class method*), 139  
`get_default_params()` (*matchzoo.models.dense\_baseline.DenseBaseline class method*), 122  
`get_default_params()` (*matchzoo.models.DenseBaseline class method*), 135  
`get_default_params()` (*matchzoo.models.DIIN class method*), 146  
`get_default_params()` (*matchzoo.models.diin.DIIN class method*), 123  
`get_default_params()` (*matchzoo.models.DRMM class method*), 137  
`get_default_params()` (*matchzoo.models.drmm.DRMM class method*), 124  
`get_default_params()` (*matchzoo.models.DRMMTKS class method*), 138  
`get_default_params()` (*matchzoo.models.drmmtk.DRMMTKS class method*), 125  
`get_default_params()` (*matchzoo.models.DSSM class method*), 135  
`get_default_params()` (*matchzoo.models.dssm.DSSM class method*), 126  
`get_default_params()` (*matchzoo.models.DUET class method*), 145  
`get_default_params()` (*matchzoo.models.duet.DUET class method*), 127  
`get_default_params()` (*matchzoo.models.ESIM class method*), 139  
`get_default_params()` (*matchzoo.models.esim.ESIM class method*), 128  
`get_default_params()` (*matchzoo.models.HBMP class method*), 145  
`get_default_params()` (*matchzoo.models.hbmp.HBMP class method*), 129  
`get_default_params()` (*matchzoo.models.KNRM class method*), 139  
`get_default_params()` (*matchzoo.models.knrm.KNRM class method*), 130  
`get_default_params()` (*matchzoo.models.knrm.KNRM class method*), 130

- zoo.models.match\_pyramid.MatchPyramid* class method), 131
- get\_default\_params()* (*matchzoo.models.match\_srrn.MatchSRNN* class method), 131
- get\_default\_params()* (*matchzoo.models.MatchLSTM* class method), 140
- get\_default\_params()* (*matchzoo.models.matchlstm.MatchLSTM* class method), 132
- get\_default\_params()* (*matchzoo.models.MatchPyramid* class method), 144
- get\_default\_params()* (*matchzoo.models.MatchSRNN* class method), 147
- get\_default\_params()* (*matchzoo.models.MVLSTM* class method), 143
- get\_default\_params()* (*matchzoo.models.mvlstm.MVLSTM* class method), 133
- get\_default\_preprocessor()* (*matchzoo.engine.base\_model.BaseModel* class method), 89
- get\_default\_preprocessor()* (*matchzoo.models.Bert* class method), 142
- get\_default\_preprocessor()* (*matchzoo.models.bert.Bert* class method), 117
- get\_default\_preprocessor()* (*matchzoo.models.CDSSM* class method), 136
- get\_default\_preprocessor()* (*matchzoo.models.cdssm.CDSSM* class method), 120
- get\_default\_preprocessor()* (*matchzoo.models.DIIN* class method), 147
- get\_default\_preprocessor()* (*matchzoo.models.diin.DIIN* class method), 123
- get\_default\_preprocessor()* (*matchzoo.models.DSSM* class method), 136
- get\_default\_preprocessor()* (*matchzoo.models.dssm.DSSM* class method), 126
- get\_default\_preprocessor()* (*matchzoo.models.DUET* class method), 145
- get\_default\_preprocessor()* (*matchzoo.models.duet.DUET* class method), 127
- get\_file()* (in module *matchzoo.utils*), 213
- get\_file()* (in module *matchzoo.utils.get\_file*), 203
- guess\_and\_fill\_missing\_params()* (*matchzoo.engine.base\_model.BaseModel* method), 88
- guess\_and\_fill\_missing\_params()* (*matchzoo.models.CDSSM* method), 137
- guess\_and\_fill\_missing\_params()* (*matchzoo.models.cdssm.CDSSM* method), 120
- ## H
- has\_label()* (*matchzoo.data\_pack.data\_pack.DataPack* property), 48
- has\_label()* (*matchzoo.data\_pack.DataPack* property), 54
- has\_label()* (*matchzoo.DataPack* property), 215
- HBMP* (class in *matchzoo.models*), 144
- HBMP* (class in *matchzoo.models.hbmp*), 129
- Histogram* (class in *matchzoo.dataloader.callbacks*), 65
- Histogram* (class in *matchzoo.dataloader.callbacks.histogram*), 59
- hyper\_space()* (*matchzoo.engine.param.Param* property), 96
- hyper\_space()* (*matchzoo.engine.param\_table.ParamTable* property), 98
- hyper\_space()* (*matchzoo.Param* property), 221
- hyper\_space()* (*matchzoo.ParamTable* property), 222
- HyperoptProxy* (class in *matchzoo.engine.hyper\_spaces*), 92

## I

*id\_left()* (*matchzoo.dataloader.DataLoader* property), 74

*id\_left()* (*matchzoo.dataloader.dataloader.DataLoader* property), 68

*is\_best\_so\_far()* (*matchzoo.utils.early\_stopping.EarlyStopping* property), 202

*is\_best\_so\_far()* (*matchzoo.utils.EarlyStopping* property), 212

## K

*keys()* (*matchzoo.engine.param\_table.ParamTable* method), 99

*keys()* (*matchzoo.ParamTable* method), 223

*KNRM* (class in *matchzoo.models*), 139

*KNRM* (class in *matchzoo.models.knrm*), 130

## L

*label()* (*matchzoo.dataloader.DataLoader* property), 74

*label()* (*matchzoo.dataloader.dataloader.DataLoader* property), 68

*LambdaCallback* (class in *matchzoo.dataloader.callbacks*), 64

*LambdaCallback* (class in *matchzoo.dataloader.callbacks.lambda\_callback*), 60

- `left()` (*matchzoo.data\_pack.data\_pack.DataPack property*), 49  
`left()` (*matchzoo.data\_pack.DataPack property*), 55  
`left()` (*matchzoo.DataPack property*), 216  
`Lemmatization` (class in *matchzoo.preprocessors.units*), 177  
`Lemmatization` (class in *matchzoo.preprocessors.units.lemmatization*), 166  
`list_available()` (in module *matchzoo.datasets*), 82  
`list_available()` (in module *matchzoo.metrics*), 113  
`list_available()` (in module *matchzoo.models*), 148  
`list_available()` (in module *matchzoo.preprocessors*), 191  
`list_available()` (in module *matchzoo.preprocessors.units*), 183  
`list_available_losses()` (*matchzoo.datasets.toy.BaseTask class method*), 80  
`list_available_losses()` (*matchzoo.engine.base\_task.BaseTask class method*), 91  
`list_available_losses()` (*matchzoo.tasks.Classification class method*), 194  
`list_available_losses()` (*matchzoo.tasks.classification.Classification class method*), 192  
`list_available_losses()` (*matchzoo.tasks.Ranking class method*), 194  
`list_available_losses()` (*matchzoo.tasks.ranking.Ranking class method*), 193  
`list_available_metrics()` (*matchzoo.datasets.toy.BaseTask class method*), 80  
`list_available_metrics()` (*matchzoo.engine.base\_task.BaseTask class method*), 92  
`list_available_metrics()` (*matchzoo.tasks.Classification class method*), 194  
`list_available_metrics()` (*matchzoo.tasks.classification.Classification class method*), 192  
`list_available_metrics()` (*matchzoo.tasks.Ranking class method*), 194  
`list_available_metrics()` (*matchzoo.tasks.ranking.Ranking class method*), 193  
`list_recursive_concrete_subclasses()` (in module *matchzoo.utils*), 209  
`list_recursive_concrete_subclasses()` (in module *matchzoo.utils.list\_recursive\_subclasses*), 204  
`load_data()` (in module *matchzoo.datasets.quora\_qp*), 78  
`load_data()` (in module *matchzoo.datasets.quora\_qp.load\_data*), 77  
`load_data()` (in module *matchzoo.datasets.snli*), 79  
`load_data()` (in module *matchzoo.datasets.snli.load\_data*), 78  
`load_data()` (in module *matchzoo.datasets.toy*), 80  
`load_data()` (in module *matchzoo.datasets.wiki\_qa*), 82  
`load_data()` (in module *matchzoo.datasets.wiki\_qa.load\_data*), 81  
`load_data_pack()` (in module *matchzoo*), 219  
`load_data_pack()` (in module *matchzoo.data\_pack*), 58  
`load_data_pack()` (in module *matchzoo.data\_pack.data\_pack*), 52  
`load_embedding()` (in module *matchzoo.datasets.toy*), 81  
`load_fasttext_embedding()` (in module *matchzoo.datasets.embeddings*), 76  
`load_fasttext_embedding()` (in module *matchzoo.datasets.embeddings.load\_fasttext\_embedding*), 75  
`load_from_file()` (in module *matchzoo.embedding*), 85  
`load_from_file()` (in module *matchzoo.embedding.embedding*), 84  
`load_glove_embedding()` (in module *matchzoo.datasets.embeddings*), 76  
`load_glove_embedding()` (in module *matchzoo.datasets.embeddings.load\_glove\_embedding*), 76  
`load_preprocessor()` (in module *matchzoo*), 219  
`load_preprocessor()` (in module *matchzoo.engine.base\_preprocessor*), 91  
`load_state_dict()` (*matchzoo.utils.early\_stopping.EarlyStopping method*), 202  
`load_state_dict()` (*matchzoo.utils.EarlyStopping method*), 212  
`loss` (in module *matchzoo.utils.parse*), 205  
`losses()` (*matchzoo.datasets.toy.BaseTask property*), 80  
`losses()` (*matchzoo.engine.base\_task.BaseTask property*), 91  
`Lowercase` (class in *matchzoo.preprocessors.units*), 177  
`Lowercase` (class in *matchzoo.preprocessors.units.lowercase*), 167

## M

`margin()` (*matchzoo.losses.rank\_hinge\_loss.RankHingeLoss*)



- property*), 101
- `margin()` (*matchzoo.losses.RankHingeLoss* property), 102
- Matching (*class in matchzoo.modules*), 160
- Matching (*class in matchzoo.modules.matching*), 153
- MatchingHistogram (*class in matchzoo.preprocessors.units*), 177
- MatchingHistogram (*class in matchzoo.preprocessors.units.matching\_histogram*), 167
- MatchingTensor (*class in matchzoo.modules*), 162
- MatchingTensor (*class in matchzoo.modules.matching\_tensor*), 154
- MatchLSTM (*class in matchzoo.models*), 140
- MatchLSTM (*class in matchzoo.models.matchlstm*), 132
- MatchModule (*class in matchzoo.modules*), 158
- MatchModule (*class in matchzoo.modules.attention*), 148
- MatchPyramid (*class in matchzoo.models*), 143
- MatchPyramid (*class in matchzoo.models.match\_pyramid*), 130
- MatchSRNN (*class in matchzoo.models*), 147
- MatchSRNN (*class in matchzoo.models.match\_srnn*), 131
- matchzoo
  - module, 33
  - matchzoo.auto
    - module, 33
    - matchzoo.auto.preparer
      - module, 33
      - matchzoo.auto.preparer.prepare
        - module, 33
      - matchzoo.auto.preparer.preparer
        - module, 34
      - matchzoo.auto.tuner
        - module, 37
        - matchzoo.auto.tuner.tune
          - module, 37
        - matchzoo.auto.tuner.tuner
          - module, 39
      - matchzoo.data\_pack
        - module, 47
        - matchzoo.data\_pack.data\_pack
          - module, 47
        - matchzoo.data\_pack.pack
          - module, 52
      - matchzoo.dataloader
        - module, 59
        - matchzoo.dataloader.callbacks
          - module, 59
          - matchzoo.dataloader.callbacks.histogram
            - module, 59
          - matchzoo.dataloader.callbacks.lambda\_callback
            - module, 60
          - matchzoo.dataloader.callbacks.ngram
            - module, 61
          - matchzoo.dataloader.callbacks.padding
            - module, 62
          - matchzoo.dataloader.dataloader
            - module, 67
            - matchzoo.dataloader.dataloader\_builder
              - module, 68
            - matchzoo.dataloader.dataset
              - module, 69
              - matchzoo.dataloader.dataset\_builder
                - module, 71
            - matchzoo.datasets
              - module, 75
              - matchzoo.datasets.embeddings
                - module, 75
                - matchzoo.datasets.embeddings.load\_fasttext\_embedding
                  - module, 75
                - matchzoo.datasets.embeddings.load\_glove\_embedding
                  - module, 76
                - matchzoo.datasets.quora\_qp
                  - module, 77
                  - matchzoo.datasets.quora\_qp.load\_data
                    - module, 77
                - matchzoo.datasets.snli
                  - module, 78
                  - matchzoo.datasets.snli.load\_data
                    - module, 78
                - matchzoo.datasets.toy
                  - module, 80
                - matchzoo.datasets.wiki\_qa
                  - module, 81
                  - matchzoo.datasets.wiki\_qa.load\_data
                    - module, 81
              - matchzoo.embedding
                - module, 83
                - matchzoo.embedding.embedding
                  - module, 83
              - matchzoo.engine
                - module, 85
                - matchzoo.engine.base\_callback
                  - module, 85
                - matchzoo.engine.base\_metric
                  - module, 86
                - matchzoo.engine.base\_model
                  - module, 87
                - matchzoo.engine.base\_preprocessor
                  - module, 90
                - matchzoo.engine.base\_task
                  - module, 91
                - matchzoo.engine.hyper\_spaces
                  - module, 92
                - matchzoo.engine.param
                  - module, 94

matchzoo.engine.param\_table  
  module, 97

matchzoo.losses  
  module, 99

matchzoo.losses.rank\_cross\_entropy\_loss  
  module, 99

matchzoo.losses.rank\_hinge\_loss  
  module, 100

matchzoo.metrics  
  module, 102

matchzoo.metrics.accuracy  
  module, 102

matchzoo.metrics.average\_precision  
  module, 103

matchzoo.metrics.cross\_entropy  
  module, 103

matchzoo.metrics.discounted\_cumulative\_gain  
  module, 104

matchzoo.metrics.mean\_average\_precision  
  module, 105

matchzoo.metrics.mean\_reciprocal\_rank  
  module, 106

matchzoo.metrics.normalized\_discounted\_cumulative\_gain  
  module, 107

matchzoo.metrics.precision  
  module, 108

matchzoo.models  
  module, 114

matchzoo.models.anmm  
  module, 114

matchzoo.models.arci  
  module, 114

matchzoo.models.arcii  
  module, 116

matchzoo.models.bert  
  module, 117

matchzoo.models.bimpm  
  module, 118

matchzoo.models.cdssm  
  module, 119

matchzoo.models.conv\_knrm  
  module, 121

matchzoo.models.dense\_baseline  
  module, 122

matchzoo.models.diin  
  module, 122

matchzoo.models.drmm  
  module, 124

matchzoo.models.drmmtks  
  module, 124

matchzoo.models.dssm  
  module, 126

matchzoo.models.duet  
  module, 127

matchzoo.models.esim  
  module, 128

matchzoo.models.hbmp  
  module, 129

matchzoo.models.knrm  
  module, 129

matchzoo.models.match\_pyramid  
  module, 130

matchzoo.models.match\_srnn  
  module, 131

matchzoo.models.matchlstm  
  module, 132

matchzoo.models.mvlstm  
  module, 132

matchzoo.models.parameter\_readme\_generator  
  module, 134

matchzoo.modules  
  module, 148

matchzoo.modules.attention  
  module, 148

matchzoo.modules.bert\_module  
  module, 149

matchzoo.modules.character\_embedding  
  module, 150

matchzoo.modules.dense\_net  
  module, 151

matchzoo.modules.dropout  
  module, 152

matchzoo.modules.gaussian\_kernel  
  module, 152

matchzoo.modules.matching  
  module, 153

matchzoo.modules.matching\_tensor  
  module, 153

matchzoo.modules.semantic\_composite  
  module, 154

matchzoo.modules.spatial\_gru  
  module, 155

matchzoo.modules.stacked\_brnn  
  module, 157

matchzoo.preprocessors  
  module, 164

matchzoo.preprocessors.basic\_preprocessor  
  module, 183

matchzoo.preprocessors.bert\_preprocessor  
  module, 185

matchzoo.preprocessors.build\_unit\_from\_data\_pack  
  module, 186

matchzoo.preprocessors.build\_vocab\_unit  
  module, 187

matchzoo.preprocessors.chain\_transform  
  module, 187

matchzoo.preprocessors.naive\_preprocessor  
  module, 188

---

|   |  |
|---|--|
| matchzoo.preprocessors.units                    | matchzoo.utils.list_recursive_subclasses                                     |
| module, 164                                     | module, 204  |
| matchzoo.preprocessors.units.character_index    | matchzoo.utils.one_hot   |
| module, 164                                     | module, 205  |
| matchzoo.preprocessors.units.digit_removal      | matchzoo.utils.parse   |
| module, 165                                     | module, 205  |
| matchzoo.preprocessors.units.frequency_filter   | matchzoo.utils.tensor_type   |
| module, 165                                     | module, 208  |
| matchzoo.preprocessors.units.lemmatization      | matchzoo.utils.timer   |
| module, 166                                     | module, 208  |
| matchzoo.preprocessors.units.lowercase          | matchzoo.version   |
| module, 167                                     | module, 214  |
| matchzoo.preprocessors.units.matching_histogram | MeanAveragePrecision (class in matchzoo.metrics), 111                        |
| matchzoo.preprocessors.units.ngram_letters      | MeanAveragePrecision (class in matchzoo.metrics.mean_average_precision), 106 |
| matchzoo.preprocessors.units.punc_removal       | MeanReciprocalRank (class in matchzoo.metrics), 111                          |
| matchzoo.preprocessors.units.stateful_unigram   | MeanReciprocalRank (class in matchzoo.metrics.mean_reciprocal_rank), 106     |
| matchzoo.preprocessors.units.stemming           | metric() (matchzoo.auto.Tuner property), 46                                  |
| module, 170                                     | metric() (matchzoo.auto.tuner.Tuner property), 42                            |
| matchzoo.preprocessors.units.stop_removal       | metric() (matchzoo.auto.tuner.tuner.Tuner property), 40                      |
| matchzoo.preprocessors.units.tokenize           | metrics() (matchzoo.datasets.toy.BaseTask property), 80                      |
| matchzoo.preprocessors.units.truncated_letters  | metrics() (matchzoo.engine.base_task.BaseTask property), 91                  |
| matchzoo.preprocessors.units.unit               | mode() (matchzoo.auto.Tuner property), 46                                    |
| module, 172                                     | mode() (matchzoo.auto.tuner.Tuner property), 42                              |
| matchzoo.preprocessors.units.vocabulary         | mode() (matchzoo.auto.tuner.tuner.Tuner property), 40                        |
| module, 172                                     | mode() (matchzoo.dataloader.Dataset property), 73                            |
| matchzoo.preprocessors.units.word_exact_match   | mode() (matchzoo.dataloader.dataset.Dataset property), 70                    |
| matchzoo.preprocessors.units.word_hashing       | module   |
| module, 174                                     | matchzoo, 33   |
| matchzoo.tasks                                  | matchzoo.auto, 33  |
| module, 191                                     | matchzoo.auto.preparer, 33   |
| matchzoo.tasks.classification                   | matchzoo.auto.preparer.prepare, 33   |
| module, 191                                     | matchzoo.auto.preparer.preparer, 34  |
| matchzoo.tasks.ranking                          | matchzoo.auto.tuner, 37  |
| module, 192                                     | matchzoo.auto.tuner.tune, 37   |
| matchzoo.trainers                               | matchzoo.auto.tuner.tuner, 39  |
| module, 195                                     | matchzoo.data_pack, 47   |
| matchzoo.trainers.trainer                       | matchzoo.data_pack.data_pack, 47   |
| module, 195                                     | matchzoo.data_pack.pack, 52  |
| matchzoo.utils                                  | matchzoo.dataloader, 59  |
| module, 200                                     | matchzoo.dataloader.callbacks, 59  |
| matchzoo.utils.average_meter                    | matchzoo.dataloader.callbacks.histogram, 59                                  |
| module, 200                                     | matchzoo.dataloader.callbacks.lambda_callback, 60                            |
| matchzoo.utils.early_stopping                   | matchzoo.dataloader.callbacks.ngram, 61                                      |
| module, 201                                     |  |
| matchzoo.utils.get_file                         |  |
| module, 202                                     |  |

- matchzoo.dataloader.callbacks.padding, 62
- matchzoo.dataloader.dataloader, 67
- matchzoo.dataloader.dataloader\_builder, 68
- matchzoo.dataloader.dataset, 69
- matchzoo.dataloader.dataset\_builder, 71
- matchzoo.datasets, 75
- matchzoo.datasets.embeddings, 75
- matchzoo.datasets.embeddings.load\_fasttext\_embeddings, 75
- matchzoo.datasets.embeddings.load\_glove\_embeddings, 76
- matchzoo.datasets.quora\_qp, 77
- matchzoo.datasets.quora\_qp.load\_data, 77
- matchzoo.datasets.snli, 78
- matchzoo.datasets.snli.load\_data, 78
- matchzoo.datasets.toy, 80
- matchzoo.datasets.wiki\_qa, 81
- matchzoo.datasets.wiki\_qa.load\_data, 81
- matchzoo.embedding, 83
- matchzoo.embedding.embedding, 83
- matchzoo.engine, 85
- matchzoo.engine.base\_callback, 85
- matchzoo.engine.base\_metric, 86
- matchzoo.engine.base\_model, 87
- matchzoo.engine.base\_preprocessor, 90
- matchzoo.engine.base\_task, 91
- matchzoo.engine.hyper\_spaces, 92
- matchzoo.engine.param, 94
- matchzoo.engine.param\_table, 97
- matchzoo.losses, 99
- matchzoo.losses.rank\_cross\_entropy\_loss, 99
- matchzoo.losses.rank\_hinge\_loss, 100
- matchzoo.metrics, 102
- matchzoo.metrics.accuracy, 102
- matchzoo.metrics.average\_precision, 103
- matchzoo.metrics.cross\_entropy, 103
- matchzoo.metrics.discounted\_cumulative\_gain, 104
- matchzoo.metrics.mean\_average\_precision, 105
- matchzoo.metrics.mean\_reciprocal\_rank, 106
- matchzoo.metrics.normalized\_discounted\_cumulative\_gain, 107
- matchzoo.metrics.precision, 108
- matchzoo.models, 114
- matchzoo.models.anmm, 114
- matchzoo.models.arci, 114
- matchzoo.models.arcii, 116
- matchzoo.models.bert, 117
- matchzoo.models.bimpm, 118
- matchzoo.models.cdssm, 119
- matchzoo.models.conv\_knrm, 121
- matchzoo.models.dense\_baseline, 122
- matchzoo.models.diin, 122
- matchzoo.models.drmm, 124
- matchzoo.models.drmm\_tks, 124
- matchzoo.models.dssm, 126
- matchzoo.models.duet, 127
- matchzoo.models.esim, 128
- matchzoo.models.hbmp, 129
- matchzoo.models.knrm, 129
- matchzoo.models.match\_pyramid, 130
- matchzoo.models.match\_srnn, 131
- matchzoo.models.match\_lstm, 132
- matchzoo.models.mvlstm, 132
- matchzoo.models.parameter\_readme\_generator, 134
- matchzoo.modules, 148
- matchzoo.modules.attention, 148
- matchzoo.modules.bert\_module, 149
- matchzoo.modules.character\_embedding, 150
- matchzoo.modules.dense\_net, 151
- matchzoo.modules.dropout, 152
- matchzoo.modules.gaussian\_kernel, 152
- matchzoo.modules.matching, 153
- matchzoo.modules.matching\_tensor, 153
- matchzoo.modules.semantic\_composite, 154
- matchzoo.modules.spatial\_gru, 155
- matchzoo.modules.stacked\_brnn, 157
- matchzoo.preprocessors, 164
- matchzoo.preprocessors.basic\_preprocessor, 183
- matchzoo.preprocessors.bert\_preprocessor, 185
- matchzoo.preprocessors.build\_unit\_from\_data\_package, 187
- matchzoo.preprocessors.build\_vocab\_unit, 187
- matchzoo.preprocessors.chain\_transform, 187
- matchzoo.preprocessors.naive\_preprocessor, 188
- matchzoo.preprocessors.units, 164
- matchzoo.preprocessors.units.character\_index, 164

matchzoo.preprocessors.units.digit\_removal, 165  
 matchzoo.preprocessors.units.frequency\_filter, 165  
 matchzoo.preprocessors.units.lemmatization, 166  
 matchzoo.preprocessors.units.lowercase, 167  
 matchzoo.preprocessors.units.matching\_histogram, 167  
 matchzoo.preprocessors.units.ngram\_letter, 168  
 matchzoo.preprocessors.units.punc\_removal, 169  
 matchzoo.preprocessors.units.stateful\_unit, 169  
 matchzoo.preprocessors.units.stemming, 170  
 matchzoo.preprocessors.units.stop\_removal, 170  
 matchzoo.preprocessors.units.tokenize, 171  
 matchzoo.preprocessors.units.truncated\_length, 171  
 matchzoo.preprocessors.units.unit, 172  
 matchzoo.preprocessors.units.vocabulary, 172  
 matchzoo.preprocessors.units.word\_exact\_match, 173  
 matchzoo.preprocessors.units.word\_hashing, 174  
 matchzoo.tasks, 191  
 matchzoo.tasks.classification, 191  
 matchzoo.tasks.ranking, 192  
 matchzoo.trainers, 195  
 matchzoo.trainers.trainer, 195  
 matchzoo.utils, 200  
 matchzoo.utils.average\_meter, 200  
 matchzoo.utils.early\_stopping, 201  
 matchzoo.utils.get\_file, 202  
 matchzoo.utils.list\_recursive\_subclasses, 204  
 matchzoo.utils.one\_hot, 205  
 matchzoo.utils.parse, 205  
 matchzoo.utils.tensor\_type, 208  
 matchzoo.utils.timer, 208  
 matchzoo.version, 214  
 mp\_matching\_func() (in module matchzoo.models.bimpm), 118  
 mp\_matching\_func\_pairwise() (in module matchzoo.models.bimpm), 119  
 MVLSTM (class in matchzoo.models), 143  
 MVLSTM (class in matchzoo.models.mvlstm), 133

**N**  
 NaivePreprocessor (class in matchzoo.preprocessors), 189  
 NaivePreprocessor (class in matchzoo.preprocessors.naive\_preprocessor), 188  
 name() (matchzoo.engine.param.Param property), 96  
 name() (matchzoo.Param property), 220  
 Ngram (class in matchzoo.dataloader.callbacks), 65  
 Ngram (class in matchzoo.dataloader.callbacks.ngram), 61  
 NgramLetter (class in matchzoo.preprocessors.units), 178  
 NgramLetter (class in matchzoo.preprocessors.units.ngram\_letter), 168  
 NormalizedDiscountedCumulativeGain (class in matchzoo.metrics), 112  
 NormalizedDiscountedCumulativeGain (class in matchzoo.metrics.normalized\_discounted\_cumulative\_gain), 107  
 num\_classes() (matchzoo.tasks.Classification property), 194  
 num\_classes() (matchzoo.tasks.classification.Classification property), 192  
 num\_dup() (matchzoo.dataloader.Dataset property), 73  
 num\_dup() (matchzoo.dataloader.dataset.Dataset property), 70  
 num\_neg() (matchzoo.dataloader.Dataset property), 73  
 num\_neg() (matchzoo.dataloader.dataset.Dataset property), 70  
 num\_neg() (matchzoo.losses.rank\_cross\_entropy\_loss.RankCrossEntropy property), 100  
 num\_neg() (matchzoo.losses.rank\_hinge\_loss.RankHingeLoss property), 100  
 num\_neg() (matchzoo.losses.RankCrossEntropyLoss property), 101  
 num\_neg() (matchzoo.losses.RankHingeLoss property), 102  
 num\_runs() (matchzoo.auto.Tuner property), 46  
 num\_runs() (matchzoo.auto.tuner.Tuner property), 42  
 num\_runs() (matchzoo.auto.tuner.tuner.Tuner property), 40

**O**  
 on\_batch\_data\_pack() (matchzoo.dataloader.callbacks.lambda\_callback.LambdaCallback method), 60  
 on\_batch\_data\_pack() (matchzoo.dataloader.callbacks.LambdaCallback method), 64

- `on_batch_data_pack()` (*matchzoo.engine.base\_callback.BaseCallback method*), 86
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.BasicPadding method*), 66
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.BertPadding method*), 66
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.DRMMPadding method*), 66
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.Histogram method*), 65
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.histogram.Histogram method*), 59
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.lambda\_callback.LambdaCallback method*), 60
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.LambdaCallback method*), 65
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.Ngram method*), 65
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.ngram.Ngram method*), 61
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.padding.BasicPadding method*), 63
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.padding.BertPadding method*), 64
  - `on_batch_unpacked()` (*matchzoo.dataloader.callbacks.padding.DRMMPadding method*), 63
  - `on_batch_unpacked()` (*matchzoo.engine.base\_callback.BaseCallback method*), 86
  - `on_epoch_end()` (*matchzoo.dataloader.Dataset method*), 72
  - `on_epoch_end()` (*matchzoo.dataloader.dataset.Dataset method*), 70
  - `one_hot()` (*in module matchzoo.utils*), 209
  - `one_hot()` (*in module matchzoo.utils.one\_hot*), 205
  - `optimizer` (*in module matchzoo.utils.parse*), 205
  - `out_channels()` (*matchzoo.modules.dense\_net.DenseNet property*), 151
  - `out_channels()` (*matchzoo.modules.DenseNet property*), 162
  - `output_dtype()` (*matchzoo.datasets.toy.BaseTask property*), 80
  - `output_dtype()` (*matchzoo.engine.base\_task.BaseTask property*), 92
  - `output_dtype()` (*matchzoo.tasks.Classification property*), 194
  - `output_dtype()` (*matchzoo.tasks.classification.Classification property*), 192
  - `output_dtype()` (*matchzoo.tasks.Ranking property*), 194
  - `output_dtype()` (*matchzoo.tasks.ranking.Ranking property*), 193
  - `output_shape()` (*matchzoo.datasets.toy.BaseTask property*), 80
  - `output_shape()` (*matchzoo.engine.base\_task.BaseTask property*), 92
  - `output_shape()` (*matchzoo.tasks.Classification property*), 194
  - `output_shape()` (*matchzoo.tasks.classification.Classification property*), 192
  - `output_shape()` (*matchzoo.tasks.Ranking property*), 194
  - `output_shape()` (*matchzoo.tasks.ranking.Ranking property*), 193
- ## P
- `pack()` (*in module matchzoo.data\_pack*), 58
  - `pack()` (*in module matchzoo.data\_pack.pack*), 52
  - `Param` (*class in matchzoo*), 219
  - `Param` (*class in matchzoo.engine.param*), 94
  - `params()` (*matchzoo.auto.Tuner property*), 46
  - `params()` (*matchzoo.auto.tuner.Tuner property*), 42
  - `params()` (*matchzoo.auto.tuner.tuner.Tuner property*), 40
  - `params()` (*matchzoo.engine.base\_model.BaseModel property*), 89
  - `ParamTable` (*class in matchzoo*), 221
  - `ParamTable` (*class in matchzoo.engine.param\_table*), 97
  - `parse_activation()` (*in module matchzoo.utils*), 210
  - `parse_activation()` (*in module matchzoo.utils.parse*), 206
  - `parse_loss()` (*in module matchzoo.utils*), 209
  - `parse_loss()` (*in module matchzoo.utils.parse*), 206
  - `parse_metric()` (*in module matchzoo.utils*), 210
  - `parse_metric()` (*in module matchzoo.utils.parse*), 207
  - `parse_optimizer()` (*in module matchzoo.utils*), 211

- `parse_optimizer()` (in module `matchzoo.utils.parse`), 207  
`Precision` (class in `matchzoo.metrics`), 109  
`Precision` (class in `matchzoo.metrics.precision`), 108  
`predict()` (`matchzoo.trainers.Trainer` method), 200  
`predict()` (`matchzoo.trainers.trainer.Trainer` method), 197  
`prepare()` (in module `matchzoo.auto.preparer`), 37  
`prepare()` (in module `matchzoo.auto.preparer.prepare`), 33  
`prepare()` (`matchzoo.auto.Preparer` method), 45  
`prepare()` (`matchzoo.auto.preparer.Preparer` method), 36  
`prepare()` (`matchzoo.auto.preparer.preparer.Preparer` method), 35  
`Preparer` (class in `matchzoo.auto`), 44  
`Preparer` (class in `matchzoo.auto.preparer`), 36  
`Preparer` (class in `matchzoo.auto.preparer.preparer`), 34  
`Progbar` (class in `matchzoo.utils.get_file`), 202  
`PuncRemoval` (class in `matchzoo.preprocessors.units`), 178  
`PuncRemoval` (class in `matchzoo.preprocessors.units.punc_removal`), 169
- ## Q
- `quuniform` (class in `matchzoo.engine.hyper_spaces`), 94
- ## R
- `RankCrossEntropyLoss` (class in `matchzoo.losses`), 101  
`RankCrossEntropyLoss` (class in `matchzoo.losses.rank_cross_entropy_loss`), 99  
`RankHingeLoss` (class in `matchzoo.losses`), 101  
`RankHingeLoss` (class in `matchzoo.losses.rank_hinge_loss`), 100  
`Ranking` (class in `matchzoo.tasks`), 194  
`Ranking` (class in `matchzoo.tasks.ranking`), 192  
`RankingMetric` (class in `matchzoo.engine.base_metric`), 87  
`relation()` (`matchzoo.data_pack.data_pack.DataPack` property), 49  
`relation()` (`matchzoo.DataPack` property), 216  
`resample()` (`matchzoo.dataloader.Dataset` property), 73  
`resample()` (`matchzoo.dataloader.dataset.Dataset` property), 70  
`resample_data()` (`matchzoo.dataloader.Dataset` method), 72  
`resample_data()` (`matchzoo.dataloader.dataset.Dataset` method), 70  
`reset()` (`matchzoo.engine.param.Param` method), 96  
`reset()` (`matchzoo.Param` method), 221  
`reset()` (`matchzoo.utils.average_meter.AverageMeter` method), 201  
`reset()` (`matchzoo.utils.AverageMeter` method), 212  
`reset()` (`matchzoo.utils.Timer` method), 212  
`reset()` (`matchzoo.utils.timer.Timer` method), 208  
`reset_index()` (`matchzoo.dataloader.Dataset` method), 72  
`reset_index()` (`matchzoo.dataloader.dataset.Dataset` method), 70  
`reset_parameters()` (`matchzoo.models.BiMPM` method), 140  
`reset_parameters()` (`matchzoo.models.bimpm.BiMPM` method), 118  
`reset_parameters()` (`matchzoo.modules.spatial_gru.SpatialGRU` method), 156  
`reset_parameters()` (`matchzoo.modules.SpatialGRU` method), 163  
`restore()` (`matchzoo.trainers.Trainer` method), 200  
`restore()` (`matchzoo.trainers.trainer.Trainer` method), 197  
`restore_model()` (`matchzoo.trainers.Trainer` method), 200  
`restore_model()` (`matchzoo.trainers.trainer.Trainer` method), 197  
`resume()` (`matchzoo.utils.Timer` method), 212  
`resume()` (`matchzoo.utils.timer.Timer` method), 208  
`right()` (`matchzoo.data_pack.data_pack.DataPack` property), 49  
`right()` (`matchzoo.data_pack.DataPack` property), 55  
`right()` (`matchzoo.DataPack` property), 216  
`RNNDropout` (class in `matchzoo.modules`), 159  
`RNNDropout` (class in `matchzoo.modules.dropout`), 152  
`run()` (`matchzoo.trainers.Trainer` method), 199  
`run()` (`matchzoo.trainers.trainer.Trainer` method), 196
- ## S
- `sample()` (in module `matchzoo.engine.hyper_spaces`), 94  
`save()` (`matchzoo.data_pack.data_pack.DataPack` method), 49  
`save()` (`matchzoo.data_pack.DataPack` method), 55  
`save()` (`matchzoo.DataPack` method), 216  
`save()` (`matchzoo.engine.base_preprocessor.BasePreprocessor` method), 91  
`save()` (`matchzoo.trainers.Trainer` method), 200  
`save()` (`matchzoo.trainers.trainer.Trainer` method), 197  
`save_model()` (`matchzoo.trainers.Trainer` method), 200

- save\_model() (*matchzoo.trainers.trainer.Trainer method*), 197
- SemanticComposite (*class in matchzoo.modules*), 161
- SemanticComposite (*class in matchzoo.modules.semantic\_composite*), 155
- set() (*matchzoo.engine.param\_table.ParamTable method*), 98
- set() (*matchzoo.ParamTable method*), 222
- set\_default() (*matchzoo.engine.param.Param method*), 96
- set\_default() (*matchzoo.Param method*), 221
- should\_stop\_early() (*matchzoo.utils.early\_stopping.EarlyStopping property*), 202
- should\_stop\_early() (*matchzoo.utils.EarlyStopping property*), 212
- shuffle() (*matchzoo.data\_pack.data\_pack.DataPack method*), 50
- shuffle() (*matchzoo.data\_pack.DataPack method*), 56
- shuffle() (*matchzoo.data\_loader.Dataset property*), 73
- shuffle() (*matchzoo.data\_loader.dataset.Dataset property*), 70
- shuffle() (*matchzoo.DataPack method*), 217
- softmax\_by\_row() (*matchzoo.modules.spatial\_gru.SpatialGRU method*), 156
- softmax\_by\_row() (*matchzoo.modules.SpatialGRU method*), 163
- sort() (*matchzoo.data\_loader.Dataset property*), 73
- sort() (*matchzoo.data\_loader.dataset.Dataset property*), 70
- sort\_and\_couple() (*in module matchzoo.engine.base\_metric*), 87
- SpaceType (*in module matchzoo.engine.param*), 94
- SpatialGRU (*class in matchzoo.modules*), 163
- SpatialGRU (*class in matchzoo.modules.spatial\_gru*), 155
- Squeeze (*class in matchzoo.models.cdssm*), 121
- StackedBRNN (*class in matchzoo.modules*), 159
- StackedBRNN (*class in matchzoo.modules.stacked\_brnn*), 157
- state() (*matchzoo.preprocessors.units.stateful\_unit.StatefulUnit property*), 169
- state() (*matchzoo.preprocessors.units.StatefulUnit property*), 179
- state\_dict() (*matchzoo.utils.early\_stopping.EarlyStopping method*), 202
- state\_dict() (*matchzoo.utils.EarlyStopping method*), 212
- StatefulUnit (*class in matchzoo.preprocessors.units*), 178
- StatefulUnit (*class in matchzoo.preprocessors.units.stateful\_unit*), 169
- Stemming (*class in matchzoo.preprocessors.units*), 179
- Stemming (*class in matchzoo.preprocessors.units.stemming*), 170
- stop() (*matchzoo.utils.Timer method*), 212
- stop() (*matchzoo.utils.timer.Timer method*), 208
- StopRemoval (*class in matchzoo.preprocessors.units*), 179
- StopRemoval (*class in matchzoo.preprocessors.units.stop\_removal*), 170
- stopwords() (*matchzoo.preprocessors.units.stop\_removal.StopRemoval property*), 171
- stopwords() (*matchzoo.preprocessors.units.StopRemoval property*), 179

## T

- TensorType (*in module matchzoo.utils*), 209
- TensorType (*in module matchzoo.utils.tensor\_type*), 208
- time() (*matchzoo.utils.Timer property*), 212
- time() (*matchzoo.utils.timer.Timer property*), 208
- Timer (*class in matchzoo.utils*), 212
- Timer (*class in matchzoo.utils.timer*), 208
- to\_frame() (*matchzoo.engine.param\_table.ParamTable method*), 98
- to\_frame() (*matchzoo.ParamTable method*), 222
- Tokenize (*class in matchzoo.preprocessors.units*), 179
- Tokenize (*class in matchzoo.preprocessors.units.tokenize*), 171
- Trainer (*class in matchzoo.trainers*), 198
- Trainer (*class in matchzoo.trainers.trainer*), 195
- trainloader() (*matchzoo.auto.Tuner property*), 46
- trainloader() (*matchzoo.auto.tuner.Tuner property*), 42
- trainloader() (*matchzoo.auto.tuner.tuner.Tuner property*), 40
- transform() (*matchzoo.engine.base\_preprocessor.BasePreprocessor method*), 90
- transform() (*matchzoo.preprocessors.basic\_preprocessor.BasicPreprocessor method*), 185
- transform() (*matchzoo.preprocessors.BasicPreprocessor method*), 191
- transform() (*matchzoo.preprocessors.bert\_preprocessor.BertPreprocessor method*), 186



`transform()` (*matchzoo.preprocessors.BertPreprocessor* method), 191  
`transform()` (*matchzoo.preprocessors.naive\_preprocessor.NaivePreprocessor* method), 188  
`transform()` (*matchzoo.preprocessors.NaivePreprocessor* method), 189  
`transform()` (*matchzoo.preprocessors.units.character\_index.CharacterIndex* method), 164  
`transform()` (*matchzoo.preprocessors.units.CharacterIndex* method), 181  
`transform()` (*matchzoo.preprocessors.units.digit\_removal.DigitRemoval* method), 165  
`transform()` (*matchzoo.preprocessors.units.DigitRemoval* method), 176  
`transform()` (*matchzoo.preprocessors.units.frequency\_filter.FrequencyFilter* method), 166  
`transform()` (*matchzoo.preprocessors.units.FrequencyFilter* method), 177  
`transform()` (*matchzoo.preprocessors.units.Lemmatization* method), 177  
`transform()` (*matchzoo.preprocessors.units.lemmatization.Lemmatization* method), 166  
`transform()` (*matchzoo.preprocessors.units.Lowercase* method), 177  
`transform()` (*matchzoo.preprocessors.units.lowercase.Lowercase* method), 167  
`transform()` (*matchzoo.preprocessors.units.matching\_histogram.MatchingHistogram* method), 168  
`transform()` (*matchzoo.preprocessors.units.MatchingHistogram* method), 178  
`transform()` (*matchzoo.preprocessors.units.ngram\_letter.NgramLetter* method), 168  
`transform()` (*matchzoo.preprocessors.units.NgramLetter* method), 178  
`transform()` (*matchzoo.preprocessors.units.punc\_removal.PuncRemoval* method), 169  
`transform()` (*matchzoo.preprocessors.units.PuncRemoval* method), 178  
`transform()` (*matchzoo.preprocessors.units.Stemming* method), 179  
`transform()` (*matchzoo.preprocessors.units.stemming.Stemming* method), 170  
`transform()` (*matchzoo.preprocessors.units.stop\_removal.StopRemoval* method), 170  
`transform()` (*matchzoo.preprocessors.units.StopRemoval* method), 179  
`transform()` (*matchzoo.preprocessors.units.Tokenize* method), 180  
`transform()` (*matchzoo.preprocessors.units.tokenize.Tokenize* method), 171  
`transform()` (*matchzoo.preprocessors.units.truncated\_length.TruncatedLength* method), 172  
`transform()` (*matchzoo.preprocessors.units.TruncatedLength* method), 183  
`transform()` (*matchzoo.preprocessors.units.Unit* method), 176  
`transform()` (*matchzoo.preprocessors.units.unit.Unit* method), 172  
`transform()` (*matchzoo.preprocessors.units.Vocabulary* method), 181  
`transform()` (*matchzoo.preprocessors.units.vocabulary.Vocabulary* method), 173  
`transform()` (*matchzoo.preprocessors.units.word\_exact\_match.WordExactMatch* method), 174  
`transform()` (*matchzoo.preprocessors.units.word\_hashing.WordHashing* method), 175  
`transform()` (*matchzoo.preprocessors.units.WordExactMatch* method), 182  
`transform()` (*matchzoo.preprocessors.units.WordHashing* method), 181  
`TruncatedLength` (class in *matchzoo.preprocessors.units*), 182  
`TruncatedLength` (class in *matchzoo.preprocessors.units.truncated\_length*), 171  
`tune()` (in module *matchzoo.auto.tuner*), 42

[tune\(\)](#) (in module `matchzoo.auto.tuner.tune`), 38  
[tune\(\)](#) (`matchzoo.auto.Tuner` method), 46  
[tune\(\)](#) (`matchzoo.auto.tuner.Tuner` method), 42  
[tune\(\)](#) (`matchzoo.auto.tuner.tuner.Tuner` method), 40  
[Tuner](#) (class in `matchzoo.auto`), 45  
[Tuner](#) (class in `matchzoo.auto.tuner`), 41  
[Tuner](#) (class in `matchzoo.auto.tuner.tuner`), 39  
[TYPE](#) (`matchzoo.datasets.toy.BaseTask` attribute), 80  
[TYPE](#) (`matchzoo.engine.base_task.BaseTask` attribute), 91  
[TYPE](#) (`matchzoo.tasks.Classification` attribute), 193  
[TYPE](#) (`matchzoo.tasks.classification.Classification` attribute), 192  
[TYPE](#) (`matchzoo.tasks.Ranking` attribute), 194  
[TYPE](#) (`matchzoo.tasks.ranking.Ranking` attribute), 193

## U

[uniform](#) (class in `matchzoo.engine.hyper_spaces`), 94  
[Unit](#) (class in `matchzoo.preprocessors.units`), 176  
[Unit](#) (class in `matchzoo.preprocessors.units.unit`), 172  
[unpack\(\)](#) (`matchzoo.data_pack.data_pack.DataPack` method), 49  
[unpack\(\)](#) (`matchzoo.data_pack.DataPack` method), 55  
[unpack\(\)](#) (`matchzoo.DataPack` method), 216  
[update\(\)](#) (`matchzoo.engine.param_table.ParamTable` method), 99  
[update\(\)](#) (`matchzoo.ParamTable` method), 223  
[update\(\)](#) (`matchzoo.utils.average_meter.AverageMeter` method), 201  
[update\(\)](#) (`matchzoo.utils.AverageMeter` method), 212  
[update\(\)](#) (`matchzoo.utils.early_stopping.EarlyStopping` method), 202  
[update\(\)](#) (`matchzoo.utils.EarlyStopping` method), 212  
[update\(\)](#) (`matchzoo.utils.get_file.Progbar` method), 203  
[USER\\_DATA\\_DIR](#) (in module `matchzoo`), 214  
[USER\\_DIR](#) (in module `matchzoo`), 214  
[USER\\_TUNED\\_MODELS\\_DIR](#) (in module `matchzoo`), 214

## V

[validate\\_context\(\)](#) (in module `matchzoo.engine.base_preprocessor`), 90  
[validate\\_file\(\)](#) (in module `matchzoo.utils.get_file`), 204  
[validator\(\)](#) (`matchzoo.engine.param.Param` property), 96  
[validator\(\)](#) (`matchzoo.Param` property), 221  
[validloader\(\)](#) (`matchzoo.auto.Tuner` property), 46  
[validloader\(\)](#) (`matchzoo.auto.tuner.Tuner` property), 42  
[validloader\(\)](#) (`matchzoo.auto.tuner.tuner.Tuner` property), 40  
[value\(\)](#) (`matchzoo.engine.param.Param` property), 96

[value\(\)](#) (`matchzoo.Param` property), 221  
[verbose\(\)](#) (`matchzoo.auto.Tuner` property), 46  
[verbose\(\)](#) (`matchzoo.auto.tuner.Tuner` property), 42  
[verbose\(\)](#) (`matchzoo.auto.tuner.tuner.Tuner` property), 40  
[Vocabulary](#) (class in `matchzoo.preprocessors.units`), 180  
[Vocabulary](#) (class in `matchzoo.preprocessors.units.vocabulary`), 172  
[Vocabulary.TermIndex](#) (class in `matchzoo.preprocessors.units`), 180  
[Vocabulary.TermIndex](#) (class in `matchzoo.preprocessors.units.vocabulary`), 173

## W

[WordExactMatch](#) (class in `matchzoo.preprocessors.units`), 182  
[WordExactMatch](#) (class in `matchzoo.preprocessors.units.word_exact_match`), 173  
[WordHashing](#) (class in `matchzoo.preprocessors.units`), 181  
[WordHashing](#) (class in `matchzoo.preprocessors.units.word_hashing`), 174