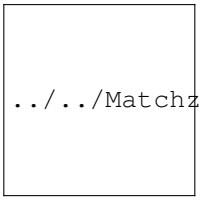
MatchZoo Documentation

发布 *2.0*

MatchZoo

2019 年 **01** 月 **12** 日

API文档



MatchZoo是一款深度文本匹配的工具。它是为了促进深度文本匹配模型的设计比较和共享而开发。这里有很多像DRMM,MatchPyramid, MV-LSTM, aNMM, DUET, ARC-I, ARC-II, DSSM和 CDSSM一样的深度匹配模型，使用统一的接口进行封装。MatchZoo主要解决的任务有文档检索，问题回答，会话应答排序，同义句识别，等等。我们非常欢迎MatchZoo的用户提出建议与意见和贡献代码。

CHAPTER 1

matchzoo package

1.1 Subpackages

1.1.1 matchzoo.auto package

Submodules

matchzoo.auto.prepare module

Prepare mode, preprocessor, and data pack.

matchzoo.auto.prepare.**prepare**(*model*, *data_pack*, *preprocessor=None*, *verbose=1*)

Prepare mode, preprocessor, and data pack.

This handles interaction among data, model, and preprocessor automatically. For example, some model like *DSSM* have dynamic input shapes based on the result of word hashing. Some models have an embedding layer which dimension is related to the data's vocabulary size. *prepare* takes care of all that and returns properly prepared model, data, and preprocessor for you.

参数

- **model** –
- **data_pack** –
- **preprocessor** – If not set, use the model's default preprocessor.
- **verbose** – Verbosity, 0 or 1.

返回类型 Tuple[*BaseModel*, *DataPack*, *BasePreprocessor*]

返回

matchzoo.auto.tuner module

Tuner class. Currently a minimum working demo.

class matchzoo.auto.tuner.**Tuner**(***kwargs*)

基类：object

Tuner.

params
return – tuner configuration paratmeters.

tune()
Tune.

Module contents

1.1.2 matchzoo.contrib package

Subpackages

matchzoo.contrib.layers package

Module contents

matchzoo.contrib.models package

Submodules

matchzoo.contrib.models.match_lstm module

Match LSTM model.

class matchzoo.contrib.models.match_lstm.**MatchLSTM**(*params=None*, *backend=None*)
基类: *matchzoo.engine.base_model.BaseModel*

Match LSTM model.

Examples

```
>>> model = MatchLSTM()
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.params['embedding_input_dim'] = 10000
>>> model.params['embedding_output_dim'] = 100
>>> model.params['embedding_trainable'] = True
>>> model.params['fc_num_units'] = 200
>>> model.params['lstm_num_units'] = 256
>>> model.params['dropout_rate'] = 0.5
>>> model.build()
```

build()
Build model.

classmethod get_default_params()
Get default parameters.

Module contents

Module contents

1.1.3 matchzoo.data_generator package

Submodules

matchzoo.data_generator.data_generator module

Base generator.

class matchzoo.data_generator.data_generator.**DataGenerator**(*data_pack*, *batch_size=32*, *shuffle=True*)

基类：keras.utils.data_utils.Sequence

Abstract base class of all matchzoo generators.

Every generator must implement _get_batch_of_transformed_samples() method.

Examples

```
>>> import matchzoo as mz
>>> raw_data = mz.datasets.toy.load_data()
>>> data_generator = DataGenerator(raw_data, batch_size=3,
...                                shuffle=False)
>>> len(data_generator)
34
>>> data_generator.num_instance
100
>>> x, y = data_generator[-1]
>>> type(x)
<class 'dict'>
>>> x.keys()
dict_keys(['id_left', 'text_left', 'id_right', 'text_right'])
>>> type(x['id_left'])
<class 'numpy.ndarray'>
>>> type(x['id_right'])
<class 'numpy.ndarray'>
>>> type(x['text_left'])
<class 'numpy.ndarray'>
>>> type(x['text_right'])
<class 'numpy.ndarray'>
>>> type(y)
<class 'numpy.ndarray'>
```

num_instance
Return the number of instances.

返回类型 int

on_epoch_end()
Reorganize the index array while epoch is ended.

reset()
Reset the generator from begin.

matchzoo.data_generator.dpool_data_generator module

Data generator with dynamic pooling.

class matchzoo.data_generator.dpool_data_generator.**DPoolDataGenerator**(*data_pack*, *fixed_length_left*, *fixed_length_right*, *compress_ratio_left=1*, *compress_ratio_right=1*, *batch_size=32*, *shuffle=True*)

基类：*matchzoo.data_generator.data_generator.DataGenerator*

Generate data with dynamic pooling.

Examples

```
>>> import matchzoo as mz
>>> raw_data = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.BasicPreprocessor(
...     fixed_length_left=10,
...     fixed_length_right=40,
...     remove_stop_words=True)
>>> processed_data = preprocessor.fit_transform(raw_data)
>>> data_generator = DPoolDataGenerator(processed_data, 3, 10,
...     batch_size=3, shuffle=False)
>>> len(data_generator)
34
>>> data_generator.num_instance
100
>>> x, y = data_generator[-1]
>>> type(x)
<class 'dict'>
>>> x.keys()
dict_keys(['id_left', 'text_left', 'length_left', 'id_right', 'text_right',
↪'length_right', 'dpool_index'])
>>> type(x['id_left'])
<class 'numpy.ndarray'>
>>> type(x['id_right'])
<class 'numpy.ndarray'>
>>> type(x['text_left'])
<class 'numpy.ndarray'>
>>> type(x['text_right'])
<class 'numpy.ndarray'>
>>> type(y)
<class 'numpy.ndarray'>
```

class matchzoo.data_generator.dpool_data_generator.**DPoolPairDataGenerator**(*data_pack*, *fixed_length_left*, *fixed_length_right*, *compress_ratio_left=1*, *compress_ratio_right=1*, *num_dup=1*, *num_neg=1*, *batch_size=32*, *shuffle=True*)

基类: *matchzoo.data_generator.pair_data_generator.PairDataGenerator*

Generate pair-wise data with dynamic pooling.

Examples

```
>>> np.random.seed(111)
>>> import matchzoo as mz
>>> raw_data = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.BasicPreprocessor(
...     fixed_length_left=10,
...     fixed_length_right=40,
...     remove_stop_words=True)
>>> processed_data = preprocessor.fit_transform(raw_data)
>>> data_generator = DPoolPairDataGenerator(processed_data, 3, 10,
...     1, 1, 2, 1, 3, False)
>>> data_generator.num_instance
10
>>> len(data_generator)
4
>>> x, y = data_generator[0]
>>> type(x)
<class 'dict'>
>>> x.keys()
dict_keys(['id_left', 'text_left', 'length_left', 'id_right', 'text_right',
↪'length_right', 'dpool_index'])
>>> type(x['id_left'])
<class 'numpy.ndarray'>
>>> type(x['id_right'])
<class 'numpy.ndarray'>
>>> type(x['text_left'])
<class 'numpy.ndarray'>
>>> type(x['text_right'])
<class 'numpy.ndarray'>
>>> len(x['id_left'])
6
>>> len(x['id_right'])
6
>>> type(y)
<class 'numpy.ndarray'>
```

matchzoo.data_generator.dynamic_data_generator module

Dynamic data generator with transform function inside.

class matchzoo.data_generator.dynamic_data_generator.**DynamicDataGenerator**(*func*, **args*, ***kwargs*)

基类：*matchzoo.data_generator.data_generator.DataGenerator*

Data generator with preprocess unit inside.

Examples

```
>>> import matchzoo as mz
>>> raw_data = mz.datasets.toy.load_data()
>>> data_generator = DynamicDataGenerator(len, data_pack=raw_data,
...                                       batch_size=1, shuffle=False)
>>> len(data_generator)
100
>>> x, y = data_generator[0]
>>> type(x)
<class 'dict'>
>>> x.keys()
dict_keys(['id_left', 'text_left', 'id_right', 'text_right'])
>>> type(x['id_left'])
<class 'numpy.ndarray'>
>>> type(x['id_right'])
<class 'numpy.ndarray'>
>>> type(x['text_left'])
<class 'numpy.ndarray'>
>>> type(x['text_right'])
<class 'numpy.ndarray'>
>>> type(y)
<class 'numpy.ndarray'>
```

matchzoo.data_generator.histogram_data_generator module

Data generator with matching histogram.

class matchzoo.data_generator.histogram_data_generator.**HistogramDataGenerator**(*data_pack*, *embedding_matrix*, *bin_size=30*, *hist_mode='C*, *batch_size=32*, *shuffle=True*)

基类：*matchzoo.data_generator.data_generator.DataGenerator*

Generate data with matching histogram.

参数

- **data_pack** (*DataPack*) – The input data pack.
- **embedding_matrix** (ndarray) – The embedding matrix used to generator match histogram.
- **bin_size** (int) – The number of bin size of the histogram.
- **hist_mode** (str) – The mode of the MatchingHistogramUnit, one of *CH*, *NH*, and *LCH*.
- **batch_size** (int) – The batch size.

- **shuffle** (bool) – Boolean, whether to shuffle the data while generating a batch.

Examples

```
>>> import matchzoo as mz
>>> raw_data = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.BasicPreprocessor()
>>> processed_data = preprocessor.fit_transform(raw_data)
>>> raw_embedding = mz.embedding.load_from_file(
...     mz.datasets.embeddings.EMBED_10_GLOVE
... )
>>> embedding_matrix = raw_embedding.build_matrix(
...     preprocessor.context['vocab_unit'].state['term_index']
... )
>>> data_generator = HistogramDataGenerator(processed_data,
...     embedding_matrix, 3, 'CH', batch_size=3, shuffle=False
... )
>>> x, y = data_generator[-1]
>>> type(x)
<class 'dict'>
>>> x.keys()
dict_keys(['id_left', 'text_left', 'length_left', 'id_right', 'text_right',
↪'length_right', 'match_histogram'])
>>> type(x['match_histogram'])
<class 'numpy.ndarray'>
>>> x['match_histogram'].shape
(1, 30, 3)
>>> type(y)
<class 'numpy.ndarray'>
```

class matchzoo.data_generator.histogram_data_generator.**HistogramPairDataGenerator**(*data_pa*
em-
bed-
ding_ma
bin_size
hist_mo
num_du
num_ne
batch_si
shuf-
fle=True

基类: *matchzoo.data_generator.pair_data_generator.PairDataGenerator*

Generate pair-wise data with matching histogram.

参数

- **data_pack** (*DataPack*) – The input data pack.
- **embedding_matrix** (ndarray) – The embedding matrix used to generator match histogram.
- **bin_size** (int) – The number of bin size of the histogram.
- **hist_mode** (str) – The mode of the MatchingHistogramUnit, one of *CH*, *NH*, and *LCH*.
- **batch_size** (int) – The batch size.
- **shuffle** (bool) – Boolean, whether to shuffle the data while generating a batch.

Examples

```
>>> np.random.seed(111)
>>> import matchzoo as mz
>>> raw_data = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.BasicPreprocessor()
>>> processed_data = preprocessor.fit_transform(raw_data)
>>> raw_embedding = mz.embedding.load_from_file(
...     mz.datasets.embeddings.EMBED_10_GLOVE
... )
>>> embedding_matrix = raw_embedding.build_matrix(
...     preprocessor.context['vocab_unit'].state['term_index']
... )
>>> data_generator = HistogramPairDataGenerator(processed_data,
...     embedding_matrix, 3, 'CH', 1, 1, 3, False)
>>> len(data_generator)
2
>>> x, y = data_generator[0]
>>> type(x)
<class 'dict'>
>>> x.keys()
dict_keys(['id_left', 'text_left', 'length_left', 'id_right', 'text_right',
↪'length_right', 'match_histogram'])
>>> type(x['match_histogram'])
<class 'numpy.ndarray'>
>>> x['match_histogram'].shape
(6, 30, 3)
>>> type(y)
<class 'numpy.ndarray'>
```

matchzoo.data_generator.histogram_data_generator.**match_histogram_generator**(*x*, *match_hist_unit*)

Generate the matching hisogram for input.

参数

- **x** (dict) – The input *dict*.
- **match_hist_unit** (*MatchingHistogramUnit*) – The histogram unit MatchingHistogramUnit.

返回类型 ndarray

返回 The matching histogram.

matchzoo.data_generator.histogram_data_generator.**trunc_text**(*input_text*, *length*)

Truncating the input text according to the input length.

参数

- **input_text** (list) – The input text need to be truncated.
- **length** (list) – The length used to truncated the text.

返回类型 list

返回 The truncated text.

matchzoo.data_generator.pair_data_generator module

Pair-wise data generator.

class matchzoo.data_generator.pair_data_generator.**PairDataGenerator**(*data_pack*, *num_dup=1*, *num_neg=1*, *batch_size=32*, *shuffle=True*)

基类: *matchzoo.data_generator.data_generator.DataGenerator*

Generate pair-wise data.

Examples

```
>>> np.random.seed(111)
>>> import matchzoo as mz
>>> raw_data = mz.datasets.toy.load_data()
>>> data_generator = PairDataGenerator(raw_data, 2, 1, 3, False)
>>> data_generator.num_instance
10
>>> len(data_generator)
4
>>> x, y = data_generator[0]
>>> type(x)
<class 'dict'>
>>> x.keys()
dict_keys(['id_left', 'text_left', 'id_right', 'text_right'])
>>> type(x['id_left'])
<class 'numpy.ndarray'>
>>> type(x['id_right'])
<class 'numpy.ndarray'>
>>> type(x['text_left'])
<class 'numpy.ndarray'>
>>> type(x['text_right'])
<class 'numpy.ndarray'>
>>> len(x['id_left'])
6
>>> len(x['id_right'])
6
>>> type(y)
<class 'numpy.ndarray'>
```

num_instance

Get the total number of pairs.

返回类型 int

classmethod reorganize_data_pack(*data_pack*, *num_dup=1*, *num_neg=1*)

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

参数

- **data_pack** (*DataPack*) – the input DataPack.
- **num_dup** (int) – number of duplicates for each positive sample.
- **num_neg** (int) – number of negative samples associated with each positive sample.

返回 the reorganized DataPack object.

Module contents

1.1.4 matchzoo.data_pack package

Submodules

matchzoo.data_pack.build_unit_from_data_pack module

Build unit from data pack.

matchzoo.data_pack.build_unit_from_data_pack.**build_unit_from_data_pack**(*unit*, *data_pack*, *mode='both'*, *flatten=True*, *verbose=1*)

Build a StatefulProcessorUnit from a DataPack object.

参数

- **unit** (*StatefulProcessorUnit*) – StatefulProcessorUnit object to be built.
- **data_pack** (*DataPack*) – The input DataPack object.
- **mode** (str) – One of 'left', 'right', and 'both', to determine the source data for building the VocabularyUnit.
- **flatten** (bool) – 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** (int) – Verbosity.

返回类型 *StatefulProcessorUnit*

返回 A built StatefulProcessorUnit object.

matchzoo.data_pack.build_vocab_unit module

Build a processor_units.VocabularyUnit given *data_pack*.

matchzoo.data_pack.build_vocab_unit.**build_vocab_unit**(*data_pack*, *mode='both'*, *verbose=1*)

Build a processor_units.VocabularyUnit given *data_pack*.

The *data_pack* should be preprocessed forehand, and each item in *text_left* and *text_right* columns of the *data_pack* should be a list of tokens.

参数

- **data_pack** (*DataPack*) – The DataPack to build vocabulary upon.
- **mode** (str) – One of 'left', 'right', and 'both', to determine the source data for building the VocabularyUnit.
- **verbose** (int) – Verbosity.

返回类型 *VocabularyUnit*

返回 A built vocabulary unit.

matchzoo.data_pack.data_pack module

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

class matchzoo.data_pack.data_pack.**DataPack**(*relation*, *left*, *right*)
基类: 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*.

参数

- **relation** (DataFrame) – Store the relation between left document and right document use ids.
- **left** (DataFrame) – Store the content or features for id_left.
- **right** (DataFrame) – 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
```

DATA_FILENAME = 'data.dill'

class FrameView(*data_pack*)
基类: object

FrameView.

append_text_length()
Append *length_left* and *length_right* columns.

参数 **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()
>>> 'length_left' in data_pack.frame[0].columns
False
>>> new_data_pack = data_pack.append_text_length()
```

(continues on next page)

(续上页)

```
>>> '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)
>>> 'length_left' in data_pack.frame[0].columns
True
```

apply_on_text (*func*, *mode='both'*, *rename=None*, *verbose=1*)

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

参数

- **func** (Callable) – The function to apply.
- **mode** (str) – One of "both", "left" and "right".
- **rename** (Optional[str]) – 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** (int) – Verbosity.

返回

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)
>>> list(frame[0].columns)
['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)
>>> list(frame[0].columns)
['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)
>>> list(frame[0].columns)
['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)
```

copy()

返回类型 *DataPack*

返回 A deep copy.

drop_label()

Remove *label* column from the data pack.

参数 **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
```

frame

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

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

返回类型 *FrameView*

返回 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
```

has_label

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

返回类型 `bool`

left

Get *left()* of *DataPack*.

返回类型 `DataFrame`

one_hot_encode_label(*num_classes=2*)

One-hot encode *label* column of *relation*.

参数

- **num_classes** – Number of classes.
- **inplace** – *True* to modify inplace, *False* to return a modified copy. (default: *False*)

返回

relation
Get *relation()* of *DataPack*.

返回类型 DataFrame

right
Get *right()* of *DataPack*.

返回类型 DataFrame

save(*dirpath*)
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*.

参数 **dirpath** (Union[str, Path]) – directory path of the saved *DataPack*.

shuffle()
Shuffle the data pack by shuffling the relation column.

参数 **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
```

unpack()
Unpack the data for training.

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

返回类型 Tuple[Dict[str, <built-in function array>], Optional[<built-in function array>]]

返回 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()
```

(continues on next page)

(续上页)

```
>>> type(y)
<class 'NoneType'>
```

matchzoo.data_pack.data_pack.**load_data_pack**(*dirpath*)

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

参数 **dirpath** (Union[str, Path]) – directory path of the saved model.

返回类型 *DataPack*

返回 a *DataPack* instance.

matchzoo.data_pack.pack module

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

matchzoo.data_pack.pack.**pack**(*df*)

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.

参数 **df** (DataFrame) – Input pandas.DataFrame to use.

Examples::

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

返回类型 *DataPack*

Module contents

1.1.5 matchzoo.datasets package

Subpackages

matchzoo.datasets.embeddings package

Submodules

matchzoo.datasets.embeddings.load_glove_embedding module

Embedding data loader.

matchzoo.datasets.embeddings.load_glove_embedding.**load_glove_embedding**(*dimension=50*)

Return the pretrained glove embedding.

参数 **dimension** (int) – the size of embedding dimension, the value can only be 50, 100, or 300.

返回类型 *Embedding*

返回 The `mz.embedding.Embedding` object.

Module contents

matchzoo.datasets.snli package

Submodules

matchzoo.datasets.snli.load_data module

SNLI data loader.

`matchzoo.datasets.snli.load_data.`**`load_data`**(*stage='train'*, *task='classification'*, *target_label='entailment'*)

Load SNLI data.

参数

- **`stage`** – One of *train*, *dev*, and *test*. (default: *train*)
- **`task`** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance. (default: *ranking*)
- **`target_label`** – If *ranking*, chose one of *entailment*, *contradiction*, *neutral*, and - as the positive label. (default: *entailment*)

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

Module contents

matchzoo.datasets.toy package

Module contents

`matchzoo.datasets.toy.`**`load_data`**(*stage='train'*, *task='ranking'*)

Load WikiQA data.

参数

- **`stage`** – One of *train*, *dev*, and *test*.
- **`task`** – Could be one of *ranking*, *classification* or a `matchzoo.engine.BaseTask` instance.

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

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.wiki_qa package

Submodules

matchzoo.datasets.wiki_qa.load_data module

WikiQA data loader.

matchzoo.datasets.wiki_qa.load_data.**load_data**(*stage='train'*, *task='ranking'*, *filter=False*)

Load WikiQA data.

参数

- **stage** – One of *train*, *dev*, and *test*.
- **task** – Could be one of *ranking*, *classification* or a matchzoo.engine.BaseTask instance.
- **filter** – Whether remove the questions without correct answers.

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

Module contents

Module contents

matchzoo.datasets.**list_available**()

1.1.6 matchzoo.engine package

Submodules

matchzoo.engine.base_metric module

Metric base class and some related utilities.

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

基类：abc.ABC

Metric base class.

ALIAS = 'base_metric'

matchzoo.engine.base_metric.**parse_metric**(*metric*)

Parse input metric in any form into a *BaseMetric* instance.

参数 **metric** (Union[str, Type[*BaseMetric*], *BaseMetric*]) – Input metric in any form.

返回 A *BaseMetric* instance

Examples::

```
>>> from matchzoo import engine, metrics
```

Use *str* as keras native metrics:

```
>>> engine.parse_metric('mse')
'mse'
```

Use *str* as MatchZoo metrics:

```
>>> mz_metric = engine.parse_metric('map')
>>> type(mz_metric)
<class 'matchzoo.metrics.mean_average_precision.MeanAveragePrecision'>
```

Use `matchzoo.engine.BaseMetric` subclasses as MatchZoo metrics:

```
>>> type(engine.parse_metric(metrics.AveragePrecision))
<class 'matchzoo.metrics.average_precision.AveragePrecision'>
```

Use `matchzoo.engine.BaseMetric` instances as MatchZoo metrics:

```
>>> type(engine.parse_metric(metrics.AveragePrecision()))
<class 'matchzoo.metrics.average_precision.AveragePrecision'>
```

matchzoo.engine.base_metric.**sort_and_couple**(*labels*, *scores*)

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

返回类型 <built-in function array>

matchzoo.engine.base_model module

Base Model.

class matchzoo.engine.base_model.**BaseModel**(*params=None*, *backend=None*)

基类：abc.ABC

Abstract base class of all matchzoo models.

BACKEND_WEIGHTS_FILENAME = 'backend_weights.h5'

PARAMS_FILENAME = 'params.dill'

backend

return model backend, a keras model instance.

返回类型 Model

build()

Build model, each sub class need to impelemnt this method.

Example

```
>>> BaseModel()
Traceback (most recent call last):
...
TypeError: Can't instantiate abstract class BaseModel ...
>>> class MyModel(BaseModel):
...     def build(self):
...         pass
>>> assert MyModel()
```

compile()

Compile model for training.

Only *keras* native metrics are compiled together with backend. MatchZoo metrics are evaluated only through *evaluate()*. Notice that *keras* count *loss* as one of the metrics while MatchZoo matchzoo.engine.BaseTask does not.

Examples

```
>>> from matchzoo import models
>>> model = models.Naive()
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.params['task'].metrics = ['mse', 'map']
>>> model.params['task'].metrics
['mse', mean_average_precision(0)]
>>> model.build()
>>> model.compile()
```

evaluate(*x*, *y*, *batch_size=128*, *verbose=1*)

Evaluate the model.

See `keras.models.Model.evaluate()` for more details.

参数

- **x** (`Union[ndarray, List[ndarray], Dict[str, ndarray]]`) – input data
- **y** (`ndarray`) – labels
- **batch_size** (`int`) – number of samples per gradient update
- **verbose** (`int`) – verbosity mode, 0 or 1

返回类型 `Dict[str, float]`

返回 scalar test loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute *model.backend.metrics_names* will give you the display labels for the scalar outputs.

Examples::

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> preprocessor = mz.preprocessors.NaivePreprocessor()
>>> data_pack = preprocessor.fit_transform(data_pack)
>>> m = mz.models.DenseBaseline()
>>> m.params['task'] = mz.tasks.Ranking()
>>> m.params['task'].metrics = [
...     'acc', 'mse', 'mae', 'ce',
...     'average_precision', 'precision', 'dcg', 'ndcg',
...     'mean_reciprocal_rank', 'mean_average_precision', 'mrr',
...     'map', 'MAP',
...     mz.metrics.AveragePrecision(threshold=1),
...     mz.metrics.Precision(k=2, threshold=2),
...     mz.metrics.DiscountedCumulativeGain(k=2),
...     mz.metrics.NormalizedDiscountedCumulativeGain(
...         k=3, threshold=-1),
...     mz.metrics.MeanReciprocalRank(threshold=2),
...     mz.metrics.MeanAveragePrecision(threshold=3)
... ]
>>> m.guess_and_fill_missing_params(verbose=0)
>>> m.build()
>>> m.compile()
>>> x, y = data_pack.unpack()
>>> evals = m.evaluate(x, y, verbose=0)
>>> type(evals)
<class 'dict'>
```

fit(*x*, *y*, *batch_size=128*, *epochs=1*, *verbose=1*, ***kwargs*)

Fit the model.

See `keras.models.Model.fit()` for more details.

参数

- **x** (Union[ndarray, List[ndarray]]) – input data.
- **y** (ndarray) – labels.
- **batch_size** (int) – number of samples per gradient update.
- **epochs** (int) – number of epochs to train the model.
- **verbose** (int) – 0, 1, or 2. Verbosity mode. 0 = silent, 1 = verbose, 2 = one log line per epoch.

Key word arguments not listed above will be propagated to keras's fit.

返回类型 History

返回 A *keras.callbacks.History* instance. Its history attribute contains all information collected during training.

fit_generator(*generator*, *epochs=1*, *verbose=1*, ***kwargs*)

Fit the model with matchzoo *generator*.

See keras.models.Model.fit_generator() for more details.

参数

- **generator** (*DataGenerator*) – A generator, an instance of engine.DataGenerator.
- **epochs** (int) – Number of epochs to train the model.
- **verbose** (int) – 0, 1, or 2. Verbosity mode. 0 = silent, 1 = verbose, 2 = one log line per epoch.

返回类型 History

返回 A *keras.callbacks.History* instance. Its history attribute contains all information collected during training.

classmethod get_default_params(*with_embedding=False*, *with_multi_layer_perceptron=False*)

Model default parameters.

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

Examples

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

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.

返回类型 *ParamTable*

返回 model parameters

classmethod get_default_preprocessor()
Model default preprocessor.

The preprocessor's transform should produce a correctly shaped data pack that can be used for training. Some extra configuration (e.g. setting *input_shapes* in `matchzoo.models.DSSMModel` may be required on the user's end.

返回类型 *BasePreprocessor*

返回 Default preprocessor.

guess_and_fill_missing_params(*verbose=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* manaully for data packs prepared for classification, then the shape of the model output and the data will mismatch.

参数 **verbose** – Verbosity.

load_embedding_matrix(*embedding_matrix*, *name='embedding'*)
Load an embedding matrix.

Load an embedding matrix into the model's embedding layer. The name of the embedding layer is specified by *name*. For models with only one embedding layer, set *name='embedding'* when creating the keras layer, and use the default *name* when load the matrix. For models with more than one embedding layers, initialize keras layer with different layer names, and set *name* accordingly to load a matrix to a chosen layer.

参数

- **embedding_matrix** (`ndarray`) – Embedding matrix to be loaded.
- **name** (`str`) – Name of the layer. (default: 'embedding')

params
return – model parameters.

返回类型 *ParamTable*

predict(*x*, *batch_size=128*)
Generate output predictions for the input samples.

See `keras.models.Model.predict()` for more details.

参数

- **x** (`Union[ndarray, List[ndarray]]`) – input data
- **batch_size** – number of samples per gradient update

返回类型 `ndarray`

返回 numpy array(s) of predictions

save(*dirpath*)
Save the model.

A saved model is represented as a directory with two files. One is a model parameters file saved by *pickle*, and the other one is a model h5 file saved by *keras*.

参数 **dirpath** (`Union[str, Path]`) – directory path of the saved model

`matchzoo.engine.base_model.`**load_model**(*dirpath*)
Load a model. The reverse function of *BaseModel.save()*.

参数 **dirpath** (`Union[str, Path]`) – directory path of the saved model

返回类型 *BaseModel*

返回 a *BaseModel* instance

matchzoo.engine.base_preprocessor module

BasePreprocessor define input and ouutput for processors.

class matchzoo.engine.base_preprocessor.**BasePreprocessor**
基类：object

BasePreprocessor to input handle data.

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

DATA_FILENAME = 'preprocessor.dill'

context
Return context.

fit(*data_pack*, *verbose=1*)
Fit parameters on input data.

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

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

参数

- **data_pack** (*DataPack*) – Datapack object to be fitted.
- **verbose** – Verbosity.

返回类型 *BasePreprocessor*

fit_transform(*data_pack*, *verbose=1*)
Call fit-transform.

参数 **data_pack** (*DataPack*) – DataPack object to be processed.

返回类型 *DataPack*

save(*dirpath*)
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*.

参数 **dirpath** (Union[str, Path]) – directory path of the saved DSSMPreprocessor.

transform(*data_pack*, *verbose=1*)
Transform input data to expected manner.

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

参数

- **data_pack** (*DataPack*) – DataPack object to be transformed.
- **verbose** – Verbosity. or list of text-left, text-right tuples.

返回类型 *DataPack*

matchzoo.engine.base_preprocessor.**load_preprocessor**(*dirpath*)
Load the fitted *context*. The reverse function of save().

参数 **dirpath** (Union[str, Path]) – directory path of the saved model.

返回类型 *DataPack*

返回 a DSSMPreprocessor instance.

matchzoo.engine.base_preprocessor.**validate_context**(*func*)
Validate context in the preprocessor.

matchzoo.engine.base_task module

Base task.

class matchzoo.engine.base_task.**BaseTask**(*loss=None*, *metrics=None*)
基类: abc.ABC

Base Task, shouldn't be used directly.

classmethod convert_metrics(*metrics*)
Convert *metrics* into properly formed list of metrics.

Examples

```
>>> BaseTask.convert_metrics(['mse'])
['mse']
>>> BaseTask.convert_metrics('map')
[mean_average_precision(0)]
```

classmethod list_available_losses()

返回类型 list

返回 a list of available losses.

classmethod list_available_metrics()

返回类型 list

返回 a list of available metrics.

loss
return – Loss used in the task.

metrics
return – Metrics used in the task.

output_dtype
return – output data type for specific task.

output_shape
return – output shape of a single sample of the task.

返回类型 tuple

matchzoo.engine.base_task.**list_available_tasks**(*base=<class 'matchzoo.engine.base_task.BaseTask'>*)

返回类型 List[Type[*BaseTask*]]

返回 a list of available task types.

matchzoo.engine.callbacks module

Callbacks.

class matchzoo.engine.callbacks.**EvaluateAllMetrics**(*model*, *x*, *y*, *once_every=1*, *batch_size=32*, *model_save_path=None*, *verbose=1*)

基类：keras.callbacks.Callback

Callback to evaluate all metrics.

MatchZoo metrics can not be evaluated batch-wise since they require dataset-level information. As a result, MatchZoo metrics are not evaluated automatically when a Model *fit*. When this callback is used, all metrics, including MatchZoo metrics and Keras metrics, are evluated once every *once_every* epochs.

参数

- **model** (*BaseModel*) – Model to evaluate.
- **x** (Union[ndarray, List[ndarray]]) –

24.

- **y** (ndarray) –

25.

- **once_every** (int) – Evaluation only triggers when *epoch % once_every == 0*. (default: 1, i.e. evaluate on every epoch's end)
- **batch_size** (int) – Number of samples per evaluation. This only affects the evaluation of Keras metrics, since MatchZoo metrics are always evaluated using the full data.
- **model_save_path** (Optional[str]) – Directory path to save the model after each evaluate callback, (default: None, i.e., no saving.)
- **verbose** – Verbosity.

on_epoch_end(*epoch*, *logs=None*)

Called at the end of en epoch.

参数

- **epoch** – integer, index of epoch.
- **logs** – dictionary of logs.

返回 dictionary of logs.

matchzoo.engine.hyper_spaces module

Hyper parameter search spaces wrapping *hyperopt*.

class matchzoo.engine.hyper_spaces.**HyperoptProxy**(*hyperopt_func*, ***kwargs*)

基类：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-refrenced 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(*name*)

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

参数 **name** (str) –

返回类型 Apply

返回 a *hyperopt* ready search space

class matchzoo.engine.hyper_spaces.**choice**(*options*)

基类: *matchzoo.engine.hyper_spaces.HyperoptProxy*

hyperopt.hp.choice() proxy.

class matchzoo.engine.hyper_spaces.**quniform**(*low*, *high*, *q=1*)

基类: *matchzoo.engine.hyper_spaces.HyperoptProxy*

hyperopt.hp.quniform() proxy.

class matchzoo.engine.hyper_spaces.**uniform**(*low*, *high*)

基类: *matchzoo.engine.hyper_spaces.HyperoptProxy*

hyperopt.hp.uniform() proxy.

matchzoo.engine.param module

Parameter class.

class matchzoo.engine.param.**Param**(*name*, *value=None*, *hyper_space=None*, *validator=None*, *desc=None*)

基类: 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'>
```

desc

return – Parameter description.

hyper_space
return – Hyper space of the parameter.

name
return – Name of the parameter.

返回类型 `str`

set_default(*val*, *verbose=1*)
Set default value, has no effect if already has a value.

参数

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

validator
return – Validator of the parameter.

返回类型 `Callable[[Any], bool]`

value
return – Value of the parameter.

返回类型 `Any`

matchzoo.engine.param_table module

Parameters table class.

class matchzoo.engine.param_table.**ParamTable**
基类: `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(*param*)

参数 **param** (*Param*) – parameter to add.

completed()

返回类型 `bool`

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

Example

```
>>> import matchzoo
>>> model = matchzoo.models.Naive()
>>> model.params.completed()
False
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.params.completed()
True
```

get(*key*)

返回类型 *Param*

返回 The parameter in the table named *key*.

hyper_space

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

返回类型 dict

keys()

返回类型 Keysview[~KT]

返回 Parameter table keys.

set(*key*, *param*)

Set *key* to parameter *param*.

Module contents

1.1.7 matchzoo.layers package

Submodules

matchzoo.layers.dynamic_pooling_layer module

An implementation of Dynamic Pooling Layer.

class matchzoo.layers.dynamic_pooling_layer.**DynamicPoolingLayer**(*psize1*, *psize2*, ***kwargs*)

基类: keras.engine.base_layer.Layer

Layer that computes dynamic pooling of one tensor.

参数

- **psize1** (int) – pooling size of dimension 1
- **psize2** (int) – pooling size of dimension 2
- **kwargs** – Standard layer keyword arguments.

Examples

```
>>> import matchzoo as mz
>>> layer = mz.layers.DynamicPoolingLayer(3, 2)
>>> num_batch, left_len, right_len, num_dim = 5, 3, 2, 10
>>> layer.build([[num_batch, left_len, right_len, num_dim],
...              [num_batch, left_len, right_len, 3]])
```

build(*input_shape*)
Build the layer.

参数 **input_shape** – the shapes of the input tensors, for DynamicPoolingLayer we need tow input tensors.

call(*inputs*, ***kwargs*)
The computation logic of DynamicPoolingLayer.

参数 **inputs** (list) – two input tensors.

返回类型 Any

compute_output_shape(*input_shape*)
Calculate the layer output shape.

参数 **input_shape** (list) – the shapes of the input tensors, for DynamicPoolingLayer we need tow input tensors.

返回类型 tuple

get_config()
Get the config dict of DynamicPoolingLayer.

返回类型 dict

matchzoo.layers.matching_layer module

An implementation of Matching Layer.

class matchzoo.layers.matching_layer.**MatchingLayer**(*normalize=False*, *matching_type='dot'*, ***kwargs*)
基类: keras.engine.base_layer.Layer

Layer that computes a matching matrix between samples in two tensors.

参数

- **normalize** (bool) – 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** (str) – the similarity function for matching
- **kwargs** – Standard layer keyword arguments.

Examples

```
>>> import matchzoo as mz
>>> layer = mz.layers.MatchingLayer(matching_type='dot',
...                                 normalize=True)
>>> num_batch, left_len, right_len, num_dim = 5, 3, 2, 10
>>> layer.build([[num_batch, left_len, num_dim],
...              [num_batch, right_len, num_dim]])
```

build(*input_shape*)
Build the layer.

参数 **input_shape** (list) – the shapes of the input tensors, for MatchingLayer we need tow input tensors.

call(*inputs*, ***kwargs*)
The computation logic of MatchingLayer.

参数 **inputs** (list) – two input tensors.

返回类型 Any

compute_output_shape(*input_shape*)
Calculate the layer output shape.

参数 **input_shape** (list) – the shapes of the input tensors, for MatchingLayer we need tow input tensors.

返回类型 tuple

get_config()
Get the config dict of MatchingLayer.

返回类型 dict

Module contents

1.1.8 matchzoo.losses package

Submodules

matchzoo.losses.rank_cross_entropy_loss module

The rank cross entropy loss.

class matchzoo.losses.rank_cross_entropy_loss.**RankCrossEntropyLoss**(*num_neg=1*)
基类: object

Rank cross entropy loss.

Examples

```
>>> from keras import backend as K
>>> softmax = lambda x: np.exp(x)/np.sum(np.exp(x), axis=0)
>>> x_pred = K.variable(np.array([[1.0], [1.2], [0.8]]))
>>> x_true = K.variable(np.array([[1], [0], [0]]))
>>> expect = -np.log(softmax(np.array([[1.0], [1.2], [0.8]])))
>>> loss = K.eval(RankCrossEntropyLoss(num_neg=2)(x_true, x_pred))
>>> np.isclose(loss, expect[0]).all()
True
```

matchzoo.losses.rank_hinge_loss module

The rank hinge loss.

class matchzoo.losses.rank_hinge_loss.**RankHingeLoss**(*num_neg=1*, *margin=1.0*)
基类: object

Rank hinge loss.

Examples

```
>>> from keras import backend as K
>>> x_pred = K.variable(np.array([[1.0], [1.2], [0.8], [1.4]]))
>>> x_true = K.variable(np.array([[1], [0], [1], [0]]))
>>> expect = ((1.0 + 1.2 - 1.0) + (1.0 + 1.4 - 0.8)) / 2
>>> expect
1.4
```

(continues on next page)

(续上页)

```
>>> loss = K.eval(RankHingeLoss(num_neg=1, margin=1.0)(x_true, x_pred))
>>> np.isclose(loss, expect)
True
```

Module contents

1.1.9 matchzoo.metrics package

Submodules

matchzoo.metrics.average_precision module

Average precision metric for ranking.

class matchzoo.metrics.average_precision.**AveragePrecision**(*threshold=0*)

基类: *matchzoo.engine.base_metric.BaseMetric*

Average precision metric.

ALIAS = ['average_precision', 'ap']

matchzoo.metrics.discounted_cumulative_gain module

Discounted cumulative gain metric for ranking.

class matchzoo.metrics.discounted_cumulative_gain.**DiscountedCumulativeGain**(*k=1*, *threshold=0*)

基类: *matchzoo.engine.base_metric.BaseMetric*

Disconunted cumulative gain metric.

ALIAS = ['discounted_cumulative_gain', 'dcg']

matchzoo.metrics.mean_average_precision module

Mean average precision metric for ranking.

class matchzoo.metrics.mean_average_precision.**MeanAveragePrecision**(*threshold=0*)

基类: *matchzoo.engine.base_metric.BaseMetric*

Mean average precision metric.

ALIAS = ['mean_average_precision', 'map']

matchzoo.metrics.mean_reciprocal_rank module

Mean reciprocal ranking metric.

class matchzoo.metrics.mean_reciprocal_rank.**MeanReciprocalRank**(*threshold=0*)

基类: *matchzoo.engine.base_metric.BaseMetric*

Mean reciprocal rank metric.

ALIAS = ['mean_reciprocal_rank', 'mrr']

matchzoo.metrics.normalized_discounted_cumulative_gain module

Normalized discounted cumulative gain metric for ranking.

class matchzoo.metrics.normalized_discounted_cumulative_gain.**NormalizedDiscountedCumulat**

基类: *matchzoo.engine.base_metric.BaseMetric*

Normalized discounted cumulative gain metric.

ALIAS = ['normalized_discounted_cumulative_gain', 'ndcg']

matchzoo.metrics.precision module

Precision for ranking.

class matchzoo.metrics.precision.**Precision**(*k=1*, *threshold=0*)

基类: *matchzoo.engine.base_metric.BaseMetric*

Precision metric.

ALIAS = 'precision'

Module contents

1.1.10 matchzoo.models package

Submodules

matchzoo.models.anmm module

An implementation of aNMM Model.

class matchzoo.models.anmm.**ANMM**(*params=None*, *backend=None*)

基类: *matchzoo.engine.base_model.BaseModel*

ANMM Model.

Examples

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

build()

Build model structure.

aNMM model based on bin weighting and query term attentions

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.arci module

An implementation of ArcI Model.

class matchzoo.models.arci.**ArcI**(*params=None*, *backend=None*)
基类: *matchzoo.engine.base_model.BaseModel*

ArcI Model.

Examples

```
>>> model = ArcI()
>>> model.params['num_blocks'] = 1
>>> 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()
```

build()
Build model structure.

ArcI use Siamese arthitecture.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.arcii module

An implementation of ArcII Model.

class matchzoo.models.arcii.**ArcII**(*params=None*, *backend=None*)
基类: *matchzoo.engine.base_model.BaseModel*

ArcII Model.

Examples

```
>>> model = ArcII()
>>> model.params['embedding_output_dim'] = 300
>>> model.params['num_blocks'] = 2
>>> 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()
```

build()
Build model structure.

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

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.cdssm module

An implementation of CDSSM (CLSM) model.

class matchzoo.models.cdssm.**CDSSM**(*params=None*, *backend=None*)
基类: *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

```
>>> model = CDSSM()
>>> model.params['optimizer'] = 'adam'
>>> model.params['filters'] =  32
>>> model.params['kernel_size'] = 3
>>> model.params['conv_activation_func'] = 'relu'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

build()
Build model structure.

CDSSM use Siamese architecture.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

classmethod get_default_preprocessor()

返回 Default preprocessor.

guess_and_fill_missing_params(*verbose=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.

参数 **verbose** – Verbosity.

matchzoo.models.conv_knrm module

ConvKNRM model.

class matchzoo.models.conv_knrm.**ConvKNRM**(*params=None*, *backend=None*)

基类: *matchzoo.models.knrm.KNRM*

ConvKNRM model.

Examples

```
>>> model = ConvKNRM()
>>> model.params['embedding_input_dim'] = 10000
>>> model.params['embedding_output_dim'] = 300
>>> model.params['embedding_trainable'] = True
>>> 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()
```

build()

Build model.

get_default_params()

Get default parameters.

matchzoo.models.dense_baseline module

A simple densely connected baseline model.

class matchzoo.models.dense_baseline.**DenseBaseline**(*params=None*, *backend=None*)

基类: *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()
>>> model.compile()
```

build()

Model structure.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.drmm module

An implementation of DRMM Model.

class matchzoo.models.drmm.**DRMM**(*params=None*, *backend=None*)
基类: *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()
>>> model.compile()
```

classmethod attention_layer(*attention_input*, *attention_mask=None*)
Performs attention on the input.

参数

- **attention_input** (Any) – The input tensor for attention layer.
- **attention_mask** (Optional[Any]) – A tensor to mask the invalid values.

返回 The masked output tensor.

build()
Build model structure.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.drmmtks module

An implementation of DRMMTKS Model.

class matchzoo.models.drmmtks.**DRMMTKS**(*params=None*, *backend=None*)
基类: *matchzoo.engine.base_model.BaseModel*

DRMMTKS Model.

Examples

```
>>> model = DRMMTKS()
>>> model.params['embedding_input_dim'] = 10000
>>> model.params['embedding_output_dim'] = 100
>>> model.params['top_k'] = 20
>>> 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 attention_layer(*attention_input*, *attention_mask=None*)
Performs attention on the input.

参数

- **attention_input** (Any) – The input tensor for attention layer.
- **attention_mask** (Optional[Any]) – A tensor to mask the invalid values.

返回 The masked output tensor.

build()
Build model structure.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.dssm module

An implementation of DSSM, Deep Structured Semantic Model.

class matchzoo.models.dssm.**DSSM**(*params=None*, *backend=None*)
基类: *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()
```

build()
Build model structure.

DSSM use Siamese arthitecture.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

classmethod get_default_preprocessor()

返回 Default preprocessor.

matchzoo.models.duet module

DUET Model.

class matchzoo.models.duet.**DUET**(*params=None*, *backend=None*)
基类: *matchzoo.engine.base_model.BaseModel*

DUET Model.

Examples

```
>>> model = DUET()
>>> model.params['embedding_input_dim'] = 1000
>>> model.params['embedding_output_dim'] = 300
>>> model.params['lm_filters'] = 32
>>> model.params['lm_hidden_sizes'] = [64, 32]
>>> model.params['dropout_rate'] = 0.5
>>> model.params['dm_filters'] = 32
>>> model.params['dm_kernel_size'] = 3
>>> model.params['dm_d_mpool'] = 4
>>> model.params['dm_hidden_sizes'] = [64, 32]
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

build()
Build model.

classmethod get_default_params()
Get default parameters.

matchzoo.models.knrm module

KNRM model.

class matchzoo.models.knrm.**KNRM**(*params=None*, *backend=None*)
基类: *matchzoo.engine.base_model.BaseModel*

KNRM model.

Examples

```
>>> model = KNRM()
>>> model.params['embedding_input_dim'] =  10000
>>> model.params['embedding_output_dim'] =  10
>>> model.params['embedding_trainable'] = 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()
```

build()
Build model.

classmethod get_default_params()
Get default parameters.

matchzoo.models.match_pyramid module

An implementation of MatchPyramid Model.

class matchzoo.models.match_pyramid.**MatchPyramid**(*params=None*, *backend=None*)
基类: *matchzoo.engine.base_model.BaseModel*

MatchPyramid Model.

Examples

```
>>> model = MatchPyramid()
>>> model.params['embedding_output_dim'] = 300
>>> model.params['num_blocks'] = 2
>>> 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()
```

build()
Build model structure.

MatchPyramid text matching as image recognition.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.mvlstm module

An implementation of MVLSTM Model.

class matchzoo.models.mvlstm.**MVLSTM**(*params=None*, *backend=None*)
基类: *matchzoo.engine.base_model.BaseModel*

MVLSTM Model.

Examples

```
>>> model = MVLSTM()
>>> model.params['lstm_units'] = 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'] = 0.5
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

build()
Build model structure.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.naive module

Naive model with a simplest structure for testing purposes.

class matchzoo.models.naive.**Naive**(*params=None*, *backend=None*)
基类: *matchzoo.engine.base_model.BaseModel*

Naive model with a simplest structure for testing purposes.

Bare minimum functioning model. The best choice to get things rolling. The worst choice to fit and evaluate performance.

build()
Build.

matchzoo.models.parameter_readme_generator module

Module contents

matchzoo.models.**list_available**()

1.1.11 matchzoo.preprocessors package

Submodules

matchzoo.preprocessors.basic_preprocessor module

Basic Preprocessor.

class matchzoo.preprocessors.basic_preprocessor.**BasicPreprocessor**(*fixed_length_left=30*, *fixed_length_right=30*, *filter_mode='df'*, *filter_low_freq=2*, *filter_high_freq=inf*, *remove_stop_words=False*)

基类: *matchzoo.engine.base_preprocessor.BasePreprocessor*

Baisc preprocessor helper.

参数

- **fixed_length_left** (int) – Integer, maximize length of left in the data_pack.
- **fixed_length_right** (int) – Integer, maximize length of right in the data_pack.
- **filter_mode** (str) – String, mode used by FrequenceFilterUnit, Can be 'df', 'cf', and 'idf'.
- **filter_low_freq** (float) – Float, lower bound value used by FrequenceFilterUnit.
- **filter_high_freq** (float) – Float, upper bound value used by FrequenceFilterUnit.
- **remove_stop_words** (bool) – 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(
...     fixed_length_left=10,
...     fixed_length_right=20,
```

(continues on next page)

(续上页)

```
...     filter_mode='df',
...     filter_low_freq=2,
...     filter_high_freq=1000,
...     remove_stop_words=True
... )
>>> preprocessor = preprocessor.fit(train_data)
>>> preprocessor.context['input_shapes']
[(10,), (20,)]
>>> preprocessor.context['vocab_size']
284
>>> processed_train_data = preprocessor.transform(train_data)
>>> type(processed_train_data)
<class 'matchzoo.data_pack.data_pack.DataPack'>
>>> test_data_transformed = preprocessor.transform(test_data)
>>> type(test_data_transformed)
<class 'matchzoo.data_pack.data_pack.DataPack'>
```

fit(*data_pack*, *verbose=1*)
Fit pre-processing context for transformation.

参数

- **data_pack** (*DataPack*) – data_pack to be preprocessed.
- **verbose** – Verbosity.

返回 class:*BasicPreprocessor* instance.

transform(*data_pack*, *verbose=1*)
Apply transformation on data, create fixed length representation.

参数

- **data_pack** (*DataPack*) – Inputs to be preprocessed.
- **verbose** – Verbosity.

返回类型 *DataPack*

返回 Transformed data as `DataPack` object.

matchzoo.preprocessors.cdssm_preprocessor module

CDSSM Preprocessor.

class matchzoo.preprocessors.cdssm_preprocessor.**CDSSMPreprocessor**(*fixed_length_left=10*, *fixed_length_right=40*, *with_word_hashing=True*)

基类: *matchzoo.engine.base_preprocessor.BasePreprocessor*

CDSSM Model preprocessor.

fit(*data_pack*, *verbose=1*)
Fit pre-processing context for transformation.

参数

- **verbose** – Verbosity.
- **data_pack** (*DataPack*) – Data_pack to be preprocessed.

返回 class:*CDSSMPreprocessor* instance.

transform(*data_pack*, *verbose=1*)
Apply transformation on data, create *letter-ngram* representation.

参数

- **data_pack** (*DataPack*) – Inputs to be preprocessed.
- **verbose** – Verbosity.

返回类型 *DataPack*

返回 Transformed data as `DataPack` object.

matchzoo.preprocessors.dssm_preprocessor module

DSSM Preprocessor.

class matchzoo.preprocessors.dssm_preprocessor.**DSSMPreprocessor**(*with_word_hashing=True*)

基类: *matchzoo.engine.base_preprocessor.BasePreprocessor*

DSSM Model preprocessor.

fit(*data_pack*, *verbose=1*)

Fit pre-processing context for transformation.

参数

- **verbose** – Verbosity.
- **data_pack** (*DataPack*) – data_pack to be preprocessed.

返回 class:*DSSMPreprocessor* instance.

transform(*data_pack*, *verbose=1*)

Apply transformation on data, create *tri-letter* representation.

参数

- **data_pack** (*DataPack*) – Inputs to be preprocessed.
- **verbose** – Verbosity.

返回类型 *DataPack*

返回 Transformed data as `DataPack` object.

matchzoo.preprocessors.naive_preprocessor module

Naive Preprocessor.

class matchzoo.preprocessors.naive_preprocessor.**NaivePreprocessor**

基类: *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)
>>> type(train_data_processed)
<class 'matchzoo.data_pack.data_pack.DataPack'>
>>> test_data_transformed = preprocessor.transform(test_data)
>>> type(test_data_transformed)
<class 'matchzoo.data_pack.data_pack.DataPack'>
```

fit(*data_pack*, *verbose=1*)

Fit pre-processing context for transformation.

参数

- **data_pack** (*DataPack*) – data_pack to be preprocessed.
- **verbose** – Verbosity.

返回 class:*NaivePreprocessor* instance.

transform(*data_pack*, *verbose=1*)

Apply transformation on data, create *tri-letter* representation.

参数

- **data_pack** (*DataPack*) – Inputs to be preprocessed.
- **verbose** – Verbosity.

返回类型 *DataPack*

返回 Transformed data as `DataPack` object.

Module contents

`matchzoo.preprocessors.`**list_available**()

1.1.12 matchzoo.processor_units package

Submodules

matchzoo.processor_units.chain_transform module

Wrapper function organizes a number of transform functions.

`matchzoo.processor_units.chain_transform.`**chain_transform**(*units*)

Compose unit transformations into a single function.

matchzoo.processor_units.processor_units module

Matchzoo toolkit for text pre-processing.

class `matchzoo.processor_units.processor_units.`**DigitRemovalUnit**

基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

Process unit to remove digits.

transform(*tokens*)

Remove digits from list of tokens.

参数 **tokens** (`list`) – list of tokens to be filtered.

Return tokens tokens of tokens without digits.

返回类型 `list`

class `matchzoo.processor_units.processor_units.`**FixedLengthUnit**(*text_length*, *pad_value=0*, *pad_mode='pre'*, *truncate_mode='pre'*)

基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

FixedLengthUnit Class.

Process unit to get the fixed length text.

Examples

```
>>> fixedlen = FixedLengthUnit(3)
>>> fixedlen.transform(range(1, 6)) == [3, 4, 5]
True
>>> fixedlen = FixedLengthUnit(3)
>>> fixedlen.transform(range(1, 3)) == [0, 1, 2]
True
```

transform(*tokens*)

Transform list of tokenized tokens into the fixed length text.

参数 **tokens** (list) – list of tokenized tokens.

Return tokens list of tokenized tokens in fixed length.

返回类型 list

class matchzoo.processor_units.processor_units.**FrequencyFilterUnit**(*low=0*, *high=inf*, *mode='df'*)

基类: *matchzoo.processor_units.processor_units.StatefulProcessorUnit*

Frequency filter unit.

参数

- **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.processor_units.FrequencyFilterUnit(
...     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.processor_units.FrequencyFilterUnit(
...     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.processor_units.FrequencyFilterUnit(
...     low=1.2, mode='idf')
>>> idf_filter.fit([['A', 'B'], ['B', 'C', 'D']])
>>> idf_filter.transform(['A', 'B', 'C'])
['A', 'C']
```

fit(*list_of_tokens*)

Fit *list_of_tokens* by calculating *mode* states.

transform(*tokens*)
Transform a list of tokens by filtering out unwanted words.

返回类型 list

class matchzoo.processor_units.processor_units.**LemmatizationUnit**
基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

Process unit for token lemmatization.

transform(*tokens*)
Lemmatization a sequence of tokens.

参数 **tokens** (list) – list of tokens to be lemmatized.

Return tokens list of lemmatizd tokens.

返回类型 list

class matchzoo.processor_units.processor_units.**LowercaseUnit**
基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

Process unit for text lower case.

transform(*tokens*)
Convert list of tokens to lower case.

参数 **tokens** (list) – list of tokens.

Return tokens lower-cased list of tokens.

返回类型 list

class matchzoo.processor_units.processor_units.**MatchingHistogramUnit**(*bin_size=30*, *embedding_matrix=None*, *normalize=True*, *mode='LCH'*)
基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

MatchingHistogramUnit Class.

参数

- **bin_size** (int) – 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** (str) – The type of the historgram, 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 = MatchingHistogramUnit(3, embedding_matrix, True, 'CH')
>>> histogram.transform([text_left, text_right])
[[3.0, 1.0, 1.0], [1.0, 2.0, 2.0]]
```

transform(*text_pair*)
Transform the input text.

返回类型 list

class matchzoo.processor_units.processor_units.**NgramLetterUnit**(*ngram=3*, *reduce_dim=True*)

基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

Process unit for n-letter generation.

Triletter is used in DSSMModel. This processor is expected to execute before *Vocab* has been created.

Examples

```
>>> triletter = NgramLetterUnit()
>>> rv = triletter.transform(['hello', 'word'])
>>> len(rv)
9
>>> rv
['#he', 'hel', 'ell', 'llo', 'lo#', '#wo', 'wor', 'ord', 'rd#']
>>> triletter = NgramLetterUnit(reduce_dim=False)
>>> rv = triletter.transform(['hello', 'word'])
>>> len(rv)
2
>>> rv
[['#he', 'hel', 'ell', 'llo', 'lo#'], ['#wo', 'wor', 'ord', 'rd#']]
```

transform(*tokens*)

Transform token into tri-letter.

For example, *word* should be represented as *#wo*, *wor*, *ord* and *rd#*.

参数 **tokens** (list) – list of tokens to be transformed.

Return n_letters generated n_letters.

返回类型 list

class matchzoo.processor_units.processor_units.**ProcessorUnit**

基类: object

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

transform(*input*)

Abstract base method, need to be implemented in subclass.

class matchzoo.processor_units.processor_units.**PuncRemovalUnit**

基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

Process unit for remove punctuations.

transform(*tokens*)

Remove punctuations from list of tokens.

参数 **tokens** (list) – list of toekns.

Return rv tokens without punctuation.

返回类型 list

class matchzoo.processor_units.processor_units.**StatefulProcessorUnit**

基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

Process unit do persive state (i.e. need fit).

fit(*input*)

Abstract base method, need to be implemented in subclass.

state
Get current state.

class matchzoo.processor_units.processor_units.**StemmingUnit**(*stemmer='porter'*)
基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

Process unit for token stemming.

transform(*tokens*)
Reducing inflected words to their word stem, base or root form.

参数

- **tokens** (list) – list of string to be stemmed.
- **stemmer** – stemmer to use, *porter* or *lancaster*.

引发 **ValueError** – stemmer type should be porter or lancaster.

Return tokens stemmed token.

返回类型 list

class matchzoo.processor_units.processor_units.**StopRemovalUnit**(*lang='english'*)
基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

Process unit to remove stop words.

Example

```
>>> unit = StopRemovalUnit()
>>> unit.transform(['a', 'the', 'test'])
['test']
>>> type(unit.stopwords)
<class 'list'>
```

stopwords
Get stopwords based on language.

Params lang language code.

返回类型 list

返回 list of stop words.

transform(*tokens*)
Remove stopwords from list of tokenized tokens.

参数

- **tokens** (list) – list of tokenized tokens.
- **lang** – language code for stopwords.

Return tokens list of tokenized tokens without stopwords.

返回类型 list

class matchzoo.processor_units.processor_units.**TokenizeUnit**
基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

Process unit for text tokenization.

transform(*input*)
Process input data from raw terms to list of tokens.

参数 **input** (str) – raw textual input.

Return tokens tokenized tokens as a list.

返回类型 list

class matchzoo.processor_units.processor_units.**VocabularyUnit**
基类: *matchzoo.processor_units.processor_units.StatefulProcessorUnit*

Vocabulary class.

Examples

```
>>> vocab = VocabularyUnit()
>>> vocab.fit(['A', 'B', 'C', 'D', 'E'])
>>> term_index = vocab.state['term_index']
>>> term_index
{'E': 1, 'C': 2, 'D': 3, 'A': 4, 'B': 5}
>>> index_term = vocab.state['index_term']
>>> index_term
{1: 'C', 2: 'A', 3: 'E', 4: 'B', 5: 'D'}
```

```
>>> term_index['out-of-vocabulary-term']
0
>>> index_term[0]
''
>>> 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, 0]
True
```

```
>>> indices = vocab.transform('ABCDDZZZ')
>>> ''.join(vocab.state['index_term'][i] for i in indices)
'ABCDD'
```

class IndexTerm
基类: dict

Map index to term.

class TermIndex
基类: dict

Map term to index.

fit(*tokens*)
Build a *TermIndex* and a *IndexTerm*.

transform(*tokens*)
Transform a list of tokens to corresponding indices.

返回类型 list

class matchzoo.processor_units.processor_units.**WordHashingUnit**(*term_index*)
基类: *matchzoo.processor_units.processor_units.ProcessorUnit*

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 = WordHashingUnit(
...     term_index={'': 0,'st#': 1, '#te': 2, 'est': 3, 'tes': 4})
>>> hashing = word_hashing.transform(letters)
>>> hashing[0]
array([0., 1., 1., 1., 1., 0.])
>>> hashing[1]
array([1., 0., 0., 0., 0., 0.])
>>> hashing.shape
(2, 6)
```

transform(*terms*)

Transform list of `letters` into word hashing layer.

参数 **terms** (`list`) – list of *tri_letters* generated by *NgramLetterUnit*.

返回类型 `ndarray`

返回 Word hashing representation of *tri-letters*.

matchzoo.processor_units.processor_units.**list_available**()

List all available units.

Module contents

1.1.13 matchzoo.tasks package

Submodules

matchzoo.tasks.classification module

Classification task.

class matchzoo.tasks.classification.**Classification**(*num_classes=2*)

基类: *matchzoo.engine.base_task.BaseTask*

Classification task.

Examples

```
>>> classification_task = Classification(num_classes=2)
>>> classification_task.metrics = ['precision']
>>> classification_task.num_classes
2
>>> classification_task.output_shape
(2,)
>>> classification_task.output_dtype
<class 'int'>
>>> print(classification_task)
Classification Task with 2 classes
```

classmethod list_available_losses()

返回类型 `list`

返回 a list of available losses.

classmethod list_available_metrics()

返回类型 list

返回 a list of available metrics.

num_classes
return – number of classes to classify.

返回类型 int

output_dtype
return – target data type, expect *int* as output.

output_shape
return – output shape of a single sample of the task.

返回类型 tuple

matchzoo.tasks.ranking module

Ranking task.

class matchzoo.tasks.ranking.**Ranking**(*loss=None*, *metrics=None*)
基类: *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
```

classmethod list_available_losses()

返回类型 list

返回 a list of available losses.

classmethod list_available_metrics()

返回类型 list

返回 a list of available metrics.

output_dtype
return – target data type, expect *float* as output.

output_shape
return – output shape of a single sample of the task.

返回类型 tuple

matchzoo.tasks.utils module

Task utilities.

matchzoo.tasks.utils.**list_available_task_types**()
Return a list of task type class objects.

返回类型 List[Type[*BaseTask*]]

Module contents

1.1.14 matchzoo.utils package

Submodules

matchzoo.utils.one_hot module

One hot vectors.

matchzoo.utils.one_hot.**one_hot**(*indices*, *num_classes*)

返回类型 ndarray

返回 A one-hot encoded vector.

matchzoo.utils.tensor_type module

Define Keras tensor type.

Module contents

1.2 Submodules

1.3 matchzoo.embedding module

Matchzoo toolkit for token embedding.

class matchzoo.embedding.**Embedding**(*data*)

基类: object

Embedding class.

Examples::

```
>>> import matchzoo as mz
>>> data_pack = mz.datasets.toy.load_data()
>>> pp = mz.preprocessors.NaivePreprocessor()
>>> vocab_unit = mz.build_vocab_unit(pp.fit_transform(data_pack),
...                                  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) + 1
True
```

To build your own:

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

build_matrix(*term_index*, *initializer=<function Embedding.<lambda>>*)
Build a matrix using *term_index*.

参数

- **term_index** (dict) – 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)*).

返回类型 ndarray

返回 A matrix.

input_dim
return Embedding input dimension.

返回类型 int

output_dim
return Embedding output dimension.

返回类型 int

matchzoo.embedding.**load_from_file**(*file_path*, *mode='word2vec'*)
Load embedding from *file_path*.

参数

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

返回类型 *Embedding*

返回 An *matchzoo.embedding.Embedding* instance.

1.4 matchzoo.logger module

MatchZoo Logging module.

1.5 matchzoo.version module

Matchzoo version file.

1.6 Module contents

CHAPTER 2

MatchZoo Model Reference

2.1 NaiveModel

2.1.1 Model Documentation

Naive model with a simplest structure for testing purposes.

Bare minimum functioning model. The best choice to get things rolling. The worst choice to fit and evaluate performance.

2.1.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.naive_model.NaiveModel'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer			choice in ['adam', 'adgrad', 'rmsprop']

2.2 DSSM

2.2.1 Model Documentation

Deep structured semantic model.

Examples:

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

2.2.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.dssm.DSSM'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer		adam	choice in ['adam', 'adgrad', 'rmsprop']
5	with_multi_layer_perceptron	A flag of whether a multiple layer perceptron is used. Shouldn't be changed.	True	
6	mlp_num_units	Number of units in first *mlp_num_layers* layers.		
7	mlp_num_layers	Number of layers of the multiple layer percetron.		
8	mlp_num_fan_out	Number of units of the layer that connects the multiple layer percetron and the output.		
9	mlp_activation_func	Activation function used in the multiple layer perceptron.		

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:

```
>>> model = CDSSM()
>>> model.params['optimizer'] = 'adam'
>>> model.params['filters'] =  32
>>> model.params['kernel_size'] = 3
>>> model.params['conv_activation_func'] = 'relu'
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

2.3.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.cdssm.CDSSM'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer			choice in ['adam', 'adgrad', 'rmsprop']
5	with_multi_layer_perceptron	A flag of whether a multiple layer perceptron is used. Shouldn't be changed.	True	
6	mlp_num_units	Number of units in first *mlp_num_layers* layers.		
7	mlp_num_layers	Number of layers of the multiple layer percetron.		
8	mlp_num_fan_out	Number of units of the layer that connects the multiple layer percetron and the output.		
9	mlp_activation_func	Activation function used in the multiple layer perceptron.		
10	filters	Number of filters in the 1D convolution layer.	32	
11	kernel_size	Number of kernel size in the 1D convolution layer.	3	
12	strides	Strides in the 1D convolution layer.	1	
13	padding	The padding mode in the convolution layer. It should be one of *same*, *valid*, and *causal*.	same	
14	conv_activation_func	Activation function in the convolution layer.	relu	
15	w_initializer		glorot_normal	
16	b_initializer		zeros	
17	dropout_rate	The dropout rate.	0.3	

2.4 DenseBaselineModel

2.4.1 Model Documentation

A simple densely connected baseline model.

Examples:

```
>>> model = DenseBaselineModel()
>>> 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()
>>> model.compile()
```

2.4.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.dense_baseline_model.DenseBaselineModel'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer			choice in ['adam', 'adgrad', 'rmsprop']
5	with_multi_layer_perceptron	A flag of whether a multiple layer perceptron is used. Shouldn't be changed.	True	
6	mlp_num_units	Number of units in first *mlp_num_layers* layers.	256	quantitative uniform distribution in [16, 512), with a step size of 1
7	mlp_num_layers	Number of layers of the multiple layer percetron.		quantitative uniform distribution in [1, 5), with a step size of 1
8	mlp_num_fan_out	Number of units of the layer that connects the multiple layer percetron and the output.		
9	mlp_activation_func	Activation function used in the multiple layer perceptron.		

2.5 ArcI

2.5.1 Model Documentation

ArcI Model.

Examples:

```
>>> model = ArcI()
>>> model.params['num_blocks'] = 1
>>> 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()
```

2.5.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.arci.ArcI'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer		adam	choice in ['adam', 'adgrad', 'rmsprop']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.	1	
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	with_multi_layer_perceptron	A flag of whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	mlp_num_units	Number of units in first *mlp_num_layers* layers.		
11	mlp_num_layers	Number of layers of the multiple layer percetron.		
12	mlp_num_fan_out	Number of units of the layer that connects the multiple layer percetron and the output.		
13	mlp_activation_func	Activation function used in the multiple layer perceptron.		
14	num_blocks	Number of convolution blocks.	1	
15	left_filters	The filter size of each convolution blocks for the left input.	[32]	
16	left_kernel_size	The kernel size of each convolution blocks for the left input.	[3]	
17	right_filters	The filter size of each convolution blocks for the right input.	[32]	
18	right_kernel_size	The kernel size of each convolution blocks for the right input.	[3]	
19	conv_activation_func	The activation function in the convolution layer.	relu	
20	left_pool_sizes	The pooling size of each convolution blocks for the left input.	[2]	
21	right_pool_sizes	The pooling size of each convolution blocks for the right input.	[2]	
22	padding	The padding mode in the convolution layer. It should be oneof *same*, *valid*, and *causal*.	same	choice in ['same', 'valid', 'causal']
23	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

2.6 ArcII

2.6.1 Model Documentation

ArcII Model.

Examples:

```
>>> model = ArcII()
>>> model.params['embedding_output_dim'] = 300
>>> model.params['num_blocks'] = 2
>>> 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()
```

2.6.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.arcii.ArcII'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer		adam	choice in ['adam', 'rmsprop', 'adagrad']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	num_blocks	Number of 2D convolution blocks.	1	
10	kernel_1d_count	Kernel count of 1D convolution layer.	32	
11	kernel_1d_size	Kernel size of 1D convolution layer.	3	
12	kernel_2d_count	Kernel count of 2D convolution layer in each block	[32]	
13	kernel_2d_size	Kernel size of 2D convolution layer in each block.	[[3, 3]]	
14	activation	Activation function.	relu	
15	pool_2d_size	Size of pooling layer in each block.	[[2, 2]]	
16	padding	The padding mode in the convolution layer. It should be oneof *same*, *valid*, and *causal*.	same	choice in ['same', 'valid', 'causal']
17	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

2.7 MatchPyramid

2.7.1 Model Documentation

MatchPyramid Model.

Examples:

```
>>> model = MatchPyramid()
>>> model.params['embedding_output_dim'] = 300
>>> model.params['num_blocks'] = 2
>>> model.params['kernel_count'] = [16, 32]
>>> model.params['kernel_size'] = [[3, 3], [3, 3]]
>>> model.params['dpool_size'] = [3, 10]
```

(continues on next page)

(续上页)

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

2.7.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.match_pyramid.MatchPyramid'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer		adam	choice in ['adam', 'rmsprop', 'adagrad']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	num_blocks	Number of convolution blocks.	1	
10	kernel_count	The kernel count of the 2D convolution of each block.	[32]	
11	kernel_size	The kernel size of the 2D convolution of each block.	[[3, 3]]	
12	activation	The activation function.	relu	
13	dpool_size	The max-pooling size of each block.	[3, 10]	
14	padding	The padding mode in the convolution layer. It should be oneof *same*, *valid*, and *causal*.	same	choice in ['same', 'valid', 'causal']
15	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.01

2.8 KNRM

2.8.1 Model Documentation

KNRM model.

Examples:

```
>>> model = KNRM()
>>> model.params['embedding_input_dim'] =  10000
>>> model.params['embedding_output_dim'] =  10
```

(continues on next page)

(续上页)

```
>>> model.params['embedding_trainable'] = 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	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.knrm.KNRM'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer			choice in ['adam', 'adgrad', 'rmsprop']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	kernel_num	The number of RBF kernels.	11	quantitative uniform distribution in [5, 20), with a step size of 1
10	sigma	The *sigma* 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 *exact_sigma* denotes the *sigma* for exact match.	0.001	

2.9 DUET

2.9.1 Model Documentation

DUET Model.

Examples:

```
>>> model = DUET()
>>> model.params['embedding_input_dim'] = 1000
>>> model.params['embedding_output_dim'] = 300
>>> model.params['lm_filters'] = 32
>>> model.params['lm_hidden_sizes'] = [64, 32]
>>> model.params['dropout_rate'] = 0.5
>>> model.params['dm_filters'] = 32
```

(continues on next page)

(续上页)

```
>>> model.params['dm_kernel_size'] = 3
>>> model.params['dm_d_mpool'] = 4
>>> model.params['dm_hidden_sizes'] = [64, 32]
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

2.9.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.duet.DUET'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer			choice in ['adam', 'adgrad', 'rmsprop']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	lm_filters	Filter size of 1D convolution layer in the local model.	32	
10	lm_hidden_sizes	A list of hidden size of the MLP layer in the local model.	[32]	
11	dm_filters	Filter size of 1D convolution layer in the distributed model.	32	
12	dm_kernel_size	Kernel size of 1D convolution layer in the distributed model.	3	
13	dm_q_hidden_size	Hidden size of the MLP layer for the left text in the distributed model.	32	
14	dm_d_mpool	Max pooling size for the right text in the distributed model.	3	
15	dm_hidden_sizes	A list of hidden size of the MLP layer in the distributed model.	[32]	
16	padding	The padding mode in the convolution layer. It should be one of *same*, *valid*, and *causal*.	same	
17	activation_func	Activation function in the convolution layer.	relu	
18	dropout_rate	The dropout rate.	0.5	quantitative uniform distribution in [0.0, 0.8), with a step size of 0.02

2.10 DRMMTKS

2.10.1 Model Documentation

DRMMTKS Model.

Examples:

```
>>> model = DRMMTKS()
>>> model.params['embedding_input_dim'] = 10000
>>> model.params['embedding_output_dim'] = 100
>>> model.params['top_k'] = 20
>>> 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()
```

2.10.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.drmmtks.DRMMTKS'>	
2	input_shapes	Dependent on the model and data. Should be set manually.	[(5,), (300,)]	
3	task	Decides model output shape, loss, and metrics.		
4	optimizer		adam	choice in ['adam', 'adgrad', 'rmsprop']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	with_multi_layer_perceptron	A flag of whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	mlp_num_units	Number of units in first *mlp_num_layers* layers.		
11	mlp_num_layers	Number of layers of the multiple layer percetron.		
12	mlp_num_fan_out	Number of units of the layer that connects the multiple layer percetron and the output.		
13	mlp_activation_func	Activation function used in the multiple layer perceptron.		
14	mask_value	The value to be masked from inputs.	-1	
15	top_k	Size of top-k pooling layer.	10	quantitative uniform distribution in [2, 100), with a step size of 1

2.11 DRMM

2.11.1 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()
>>> model.compile()
```

2.11.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.drmm.DRMM'>	
2	input_shapes	Dependent on the model and data. Should be set manually.	[(5,), (5, 30)]	
3	task	Decides model output shape, loss, and metrics.		
4	optimizer		adam	choice in ['adam', 'adgrad', 'rmsprop']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	with_multi_layer_perceptron	A flag of whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	mlp_num_units	Number of units in first *mlp_num_layers* layers.		
11	mlp_num_layers	Number of layers of the multiple layer percetron.		
12	mlp_num_fan_out	Number of units of the layer that connects the multiple layer percetron and the output.		
13	mlp_activation_func	Activation function used in the multiple layer perceptron.		
14	mask_value	The value to be masked from inputs.	-1	

2.12 ANMM

2.12.1 Model Documentation

ANMM Model.

Examples:

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

2.12.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.anmm.ANMM'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer			choice in ['adam', 'adgrad', 'rmsprop']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	dropout_rate	The dropout rate.	0.1	
10	num_layers	Number of hidden layers in the MLP layer.	2	
11	hidden_sizes	Number of hidden size for each hidden layer	[30, 30]	

2.13 MVLSTM

2.13.1 Model Documentation

MVLSTM Model.

Examples:

```
>>> model = MVLSTM()
>>> model.params['lstm_units'] = 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'] = 0.5
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.build()
```

2.13.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.models.mvlstm.MVLSTM'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer		adam	choice in ['adam', 'adgrad', 'rmsprop']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	with_multi_layer_perceptron	A flag of whether a multiple layer perceptron is used. Shouldn't be changed.	True	
10	mlp_num_units	Number of units in first *mlp_num_layers* layers.		
11	mlp_num_layers	Number of layers of the multiple layer percetron.		
12	mlp_num_fan_out	Number of units of the layer that connects the multiple layer percetron and the output.		
13	mlp_activation_func	Activation function used in the multiple layer perceptron.		
14	lstm_units	Integer, the hidden size in the bidirectional LSTM layer.	32	
15	dropout_rate	Float, the dropout rate.	0.0	
16	top_k	Integer, the size of top-k pooling layer.	10	quantitative uniform distribution in [2, 100), with a step size of 1

2.14 MatchLSTM

2.14.1 Model Documentation

Match LSTM model.

Examples:

```
>>> model = MatchLSTM()
>>> model.guess_and_fill_missing_params(verbose=0)
>>> model.params['embedding_input_dim'] = 10000
>>> model.params['embedding_output_dim'] = 100
>>> model.params['embedding_trainable'] = True
>>> model.params['fc_num_units'] = 200
```

(continues on next page)

(续上页)

```
>>> model.params['lstm_num_units'] = 256
>>> model.params['dropout_rate'] = 0.5
>>> model.build()
```

2.14.2 Model Hyper Parameters

	Name	Description	Default Value	Default Hyper-Space
0	name	Not related to the model's behavior.		
1	model_class	Model class. Used internally for save/load. Changing this may cause unexpected behaviors.	<class 'matchzoo.contrib.models.match_lstm.MatchLSTM'>	
2	input_shapes	Dependent on the model and data. Should be set manually.		
3	task	Decides model output shape, loss, and metrics.		
4	optimizer			choice in ['adam', 'adgrad', 'rmsprop']
5	with_embedding	A flag used help *auto* module. Shouldn't be changed.	True	
6	embedding_input_dim	Usually equals vocab size + 1. Should be set manually.		
7	embedding_output_dim	Should be set manually.		
8	embedding_trainable	*True* to enable embedding layer training, *False* to freeze embedding parameters.		
9	lstm_num_units	The hidden size in the LSTM layer.	256	quantitative uniform distribution in [128, 384), with a step size of 32
10	fc_num_units	The hidden size in the full connection layer.	200	quantitative uniform distribution in [100, 300), with a step size of 20
11	dropout_rate	The dropout rate.	0.0	quantitative uniform distribution in [0.0, 0.9), with a step size of 0.01

CHAPTER 3

Indices and tables

- genindex
- modindex
- 搜索

Python 模块索引

m

索引

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W