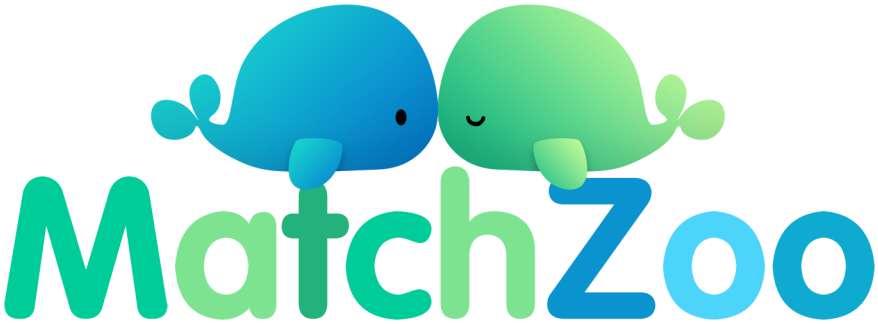
MatchZoo Documentation

发布 *2.0*

MatchZoo

2019 年 **01** 月 **09** 日

API文档



MatchZoo是一个用于文本匹配的工具包。它的开发重点是促进深文本匹配模型的设计，比较和共享。这里有许多深度匹配的方法，如DRMM，MatchPyramid，MV-LSTM，aNMM，DUET，ARC-I，ARC-II，DSSM和CDSSM，采用统一的接口设计与MatchZoo相关的潜在任务包括文档检索，问题回答，会话响应排名，同义句识别等。我们很乐意接收来自所有MatchZoo用户的任何代码贡献，建议和评论。

CHAPTER 1

matchzoo package

1.1 Subpackages

1.1.1 matchzoo.engine package

Submodules

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_FILENAME = 'backend.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
```

(continues on next page)

(续上页)

```
>>> MyModel
<class 'matchzoo.engine.base_model.MyModel'>
```

compile()
Compile model for training.

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]]`) – 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 `Union[float, List[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.

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

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()
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()
```

(continues on next page)

(续上页)

```
...         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

guess_and_fill_missing_params()

Guess and fill missing parameters in *params*.

Note: likely to be moved to a higher level API in the future.

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

Base Preprocessor, consist of multiple `ProcessorUnit`.

Each sub-class should employ a sequence of `ProcessorUnit` and `StatefulProcessorUnit` to handle input data.

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

基类: `object`

Abstract base class for model-wise processors.

fit_transform(*text_left*, *text_right*, *labels*)
Apply fit-transform on input data.

This method is an abstract method, need to be implemented in sub-class.

handle(*process_unit*, *input*)
Inference whether a process_unit is *Stateful*.

参数

- **process_unit** (*ProcessorUnit*) – Given a process unit instance.
- **input** (Any) – process input text.

Return ctx Context as dict, i.e. fitted parameters.

返回类型 Union[dict, Any]

返回 Transformed user input given transformer.

matchzoo.engine.base_task module

Base task.

class matchzoo.engine.base_task.**BaseTask**
基类: abc.ABC

Base Task, shouldn't be used directly.

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_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.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.

Example

```
>>> from hyperopt.pyll.stochastic import sample
>>> numbers = [0, 1, 2]
>>> sample(choice(options=numbers)('numbers')) in numbers
True
>>> 0 <= sample(quniform(low=0, high=9)('digit')) <= 9
True
```

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*)

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):
```

(continues on next page)

(续上页)

```
    ...
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
<hyperopt.pyll.base.Apply object at 0x...>
>>> from hyperopt.pyll.stochastic import sample
>>> samples = [sample(param.hyper_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'>
```

hyper_space

return – Hyper space of the parameter.

返回类型 `Apply`

name

return – Name of the parameter.

返回类型 `str`

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.NaiveModel()
>>> model.params.completed()
False
>>> model.guess_and_fill_missing_params()
>>> model.params.completed()
True
```

hyper_space

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

返回类型 dict

matchzoo.engine.tune module

Tuner class. Currently a minimum working demo.

matchzoo.engine.tune.**tune**(*model*, *max_evals=32*)

Tune the *model max_evals* times.

Construct a hyper parameter searching space by extracting all parameters in *model* that have a defined hyper space. Then, using *hyperopt* API, iteratively sample parameters and test for loss, and pick the best trial out of all. Currently a minimum working demo.

参数

- **model** (*BaseModel*) –
- **max_evals** (int) – Number of evaluations of a single tuning process.

返回类型 list

返回 A list of trials of the tuning process.

Example

```
>>> from matchzoo.models import DenseBaselineModel
>>> model = DenseBaselineModel()
>>> max_evals = 4
>>> trials = tune(model, max_evals)
>>> len(trials) == max_evals
True
```

Module contents

1.1.2 matchzoo.models package

Submodules

matchzoo.models.dense_baseline_model module

A simple densely connected baseline model.

class matchzoo.models.dense_baseline_model.**DenseBaselineModel**(*params=None*, *backend=None*)

基类: *matchzoo.engine.base_model.BaseModel*

A simple densely connected baseline model.

Examples

```
>>> model = DenseBaselineModel()
>>> model.params['input_shapes'] = [(30,), (30,)]
>>> model.params['num_dense_units'] = 1024
>>> model.guess_and_fill_missing_params()
>>> model.build()
```

build()

Model structure.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.dssm_model module

An implementation of DSSM, Deep Structured Semantic Model.

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

Deep structured semantic model.

Examples

```
>>> model = DSSMModel()
>>> model.guess_and_fill_missing_params()
>>> model.build()
```

build()
Build model structure.

DSSM use Siamese arthitecture.

classmethod get_default_params()

返回类型 *ParamTable*

返回 model default parameters.

matchzoo.models.naive_model module

Naive model with a simplest structure for testing purposes.

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

Naive model with a simplest structure for testing purposes.

build()
Build.

Module contents

1.1.3 matchzoo.preprocessor package

Submodules

matchzoo.preprocessor.process_units module

Matchzoo toolkit for text pre-processing.

class matchzoo.preprocessor.process_units.**DigitRemovalUnit**
基类: *matchzoo.preprocessor.process_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.preprocessor.process_units.**LemmatizationUnit**
基类: *matchzoo.preprocessor.process_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.preprocessor.process_units.**LowercaseUnit**
基类: *matchzoo.preprocessor.process_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.preprocessor.process_units.**NgramLetterUnit**
基类: *matchzoo.preprocessor.process_units.StatefulProcessorUnit*

Process unit for n-letter generation.

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

Returned *input_dim* is the dimensionality of DSSMModel.

fit(*tokens*, *ngram=3*)
Fiitting parameters (shape of word hashing layer) for :DSSM:.

参数

- **tokens** (list) – list of tokens to be fitted.
- **ngram** (int) – By default use 3-gram (tri-letter).

transform(*tokens*, *ngram=3*)
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.
- **ngram** (int) – By default use 3-gram (tri-letter).

返回类型 list

返回 set of tri-letters, dependent on *ngram*.

class matchzoo.preprocessor.process_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.preprocessor.process_units.**PuncRemovalUnit**
基类: *matchzoo.preprocessor.process_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.preprocessor.process_units.**StatefulProcessorUnit**
基类: *matchzoo.preprocessor.process_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.preprocessor.process_units.**StemmingUnit**(*stemmer='porter'*)
基类: *matchzoo.preprocessor.process_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.preprocessor.process_units.**StopRemovalUnit**(*lang='en'*)
基类: *matchzoo.preprocessor.process_units.ProcessorUnit*

Process unit to remove stop words.

get_stopwords()
Get stopwords based on language.

Params lang language code.

Return stop_list list of stop words.

返回类型 list

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.preprocessor.process_units.**TokenizeUnit**
基类: *matchzoo.preprocessor.process_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.preprocessor.process_units.**VocabularyUnit**

基类: *matchzoo.preprocessor.process_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

Module contents

1.1.4 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.

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_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**

基类: *matchzoo.engine.base_task.BaseTask*

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_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

1.2 Submodules

1.3 matchzoo.datapack module

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

class matchzoo.datapack.**DataPack**(*data*, *context={}*)
基类：object

Matchzoo DataPack data structure, store dataframe and context.

Example

```
>>> features = [([1,3], [2,3]), ([3,0], [1,6])]
>>> context = {'vocab_size': 2000}
>>> dp = DataPack(data=features,
...               context=context)
>>> type(dp.sample(1))
<class 'matchzoo.datapack.DataPack'>
>>> len(dp)
2
>>> features, context = dp.dataframe, dp.context
>>> context
{'vocab_size': 2000}
```

DATA_FILENAME = 'data.dill'

append(*other*)
Append a new *DataPack* object to current *DataPack* object.

It should be noted that the context of the previous *DataPack* will be updated by the new one.

参数 **other** (*DataPack*) – the *DataPack* object to be appended.

context
Get context of *DataPack*.

dataframe
Get data frame.

sample(*number*, *replace=True*)
Sample records from *DataPack* object, for generator.

参数

- **number** – number of records to be sampled, use *batch_size*.
- **replace** – sample with replacement, default value is *True*.

Return data_pack return *DataPack* object including sampled data and context (shallow copy of the context‘).

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*.

matchzoo.datapack.**load_datapack**(*dirpath*)
Load a *DataPack*. The reverse function of *DataPack.save()*.

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

返回类型 *DataPack*

返回 a *DataPack* instance

1.4 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- 搜索

Python 模块索引

m

索引

L

M

N

O

P

Q

R

S

T

U

V