
longling Documentation

alpha

tongshiwei

2022 04 26

1	Overview	3
2	Quick scripts	5
2.1	Neural Network	5
2.2	CLI	5
3	Module Index	7
3.1	Command Line Interfaces	7
3.2	General Library	9
3.3	Spider	32
3.4	Architecture Tools for Constructing Projects	32
3.5	Machine Learning	35
Python		63
		65



CHAPTER 1

Overview

The project contains several modules for different purposes:

- **lib** serves as the basic toolkit that can be used in any place without extra dependencies.
- **ML** provides many interfaces to quickly build machine learning tools.

The project provide several cli scripts to help construct different architecture.

2.1 Neural Network

- [glue](#) for mxnet-gluon

2.2 CLI

Provide several general tools, consistently invoked by:

```
longling $subcommand $parameters1 $parameters2
```

To see the help information:

```
longling -- --help
longling $subcommand --help
```

Take a glance on [all available cli](#).

The cli tools is constructed based on [fire](#). Refer to the [documentation](#) for detailed usage.

2.2.1 Demo

Split dataset

target: split a dataset into train/valid/test

```
longling train_valid_test $filename1 $filename2 -- --train_ratio 0.8 --valid_ratio 0.
↪1 --test_ratio 0.1
```

Similar commands:

- train_test

```
longling train_test $filename1 -- --train_ratio 0.8 --test_ratio 0.2
```

- train_valid

```
longling train_valid $filename1 -- --train_ratio 0.8 --valid_ratio 0.2
```

- Cross Validation kfold

```
longling kfold $filename1 $filename2 -- --n_splits 5
```

Display the tree of content

```
longling toc .
```

such as

```
/
├── __init__.py
├── __pycache__/
│   ├── __init__.cpython-36.pyc
│   └── toc.cpython-36.pyc
└── toc.py
```

Quickly construct a project

```
longling arch
```

or you can also directly copy the template files

```
longling arch-cli
```

To be noticed that, you need to check \$VARIABLE in the template files.

Result Analysis

The cli tools for result analysis is specially designed for json result format:

```
longling max $filename $key1 $key2 $key3
longling amax $key1 $key2 $key3 --src $filename
```

For the composite key like {'prf':{'avg':{'f1': 0.77}}}, the key should be presented as prf:avg:f1. Thus, all the key used in the result file should not contain :.

3.1 Command Line Interfaces

use `longling -- --help` to see all available cli, and use `longling $subcommand -- --help` to see concrete help information for a certain cli, e.g. `longling encode -- --help`

3.1.1 Format and Encoding

<code>longling.lib.stream.encode(src, ...)</code>	Convert a file in source encoding to target encoding
<code>longling.lib.loading.csv2jsonl(src, ...[, csv /io json /io ...])</code>	
<code>longling.lib.loading.jsonl2csv(src, ...[, json /io csv /io ...])</code>	

3.1.2 Download Data

<code>longling.spider.download_data.download_file(url)</code>	cli alias: download, download data from specified url
---	---

3.1.3 Architecture

<code>longling.toolbox.toc.toc([root, parent, ...])</code>	
<code>longling.Architecture.cli.cli([skip_top, ...])</code>	The main function for arch
<code>longling.Architecture.install_file.nni([tar_dir])</code>	cli alias: arch nni and install nni

3 –

<code>longling.Architecture.install_file.gitignore(...)</code>	cli alias: arch gitignore
<code>longling.Architecture.install_file.pytest(...)</code>	cli alias: arch pytest
<code>longling.Architecture.install_file.coverage(...)</code>	cli alias: arch coverage
<code>longling.Architecture.install_file.pysetup([...])</code>	cli alias: arch pysetup
<code>longling.Architecture.install_file.sphinx_conf([...])</code>	cli alias: arch sphinx_conf
<code>longling.Architecture.install_file.makefile([...])</code>	cli alias: arch makefile
<code>longling.Architecture.install_file.readthedocs([...])</code>	cli alias: arch readthedocs
<code>longling.Architecture.install_file.travis(...)</code>	cli alias: arch travis
<code>longling.Architecture.install_file.dockerfile(atype)</code>	cli alias: arch dockerfile
<code>longling.Architecture.install_file.gitlab_ci(...)</code>	cli alias: arch gitlab_ci
<code>longling.Architecture.install_file.chart(...)</code>	cli alias: arch chart

3.1.4 Model Selection

Validation on Datasets

Split dataset to train, valid and test or apply kfold.

<code>longling.ML.toolkit.dataset.train_valid_test(...)</code>	param files
<code>longling.ML.toolkit.dataset.train_test(...)</code>	param files
<code>longling.ML.toolkit.dataset.kfold(*files[, ...])</code>	

Select Best Model

Select best models on specified conditions

<code>longling.ML.toolkit.analyser.cli.select_max(...)</code>	cli alias: max
<code>longling.ML.toolkit.analyser.cli.arg_select_max(...)</code>	cli alias: amax
<code>longling.ML.toolkit.hyper_search.nni.show_top_k(k)</code>	Updated in v1.3.17

<code>longling.ML.toolkit.hyper_search.nni.show(key)</code>	Updated in v1.3.17
---	--------------------

3.2 General Library

3.2.1 Quick Glance

For io:

<code>longling.lib.stream.to_io(stream, <class >, ...)</code>	Convert an object as an io stream, could be a path to file or an io stream.
<code>longling.lib.stream.as_io(src, <class >, ...)</code>	with wrapper for to_io function, default mode is "r"
<code>longling.lib.stream.as_out_io(tar, <class >, ...)</code>	with wrapper for to_io function, default mode is "w"
<code>longling.lib.loading.loading(src, <class >), ...)</code>	

<code>longling.lib.iterator.AsyncLoopIter(src[, ...])</code>	,
<code>longling.lib.iterator.CacheAsyncLoopIter(...)</code>	
<code>longling.lib.iterator.iterwrap(itertype, ...)</code>	AsyncLoopIter

<code>longling.lib.utilog.config_logging([...])</code>	
--	--

For path

<code>longling.lib.path.path_append(path, *addition)</code>	
<code>longling.lib.path.abs_current_dir(filepath)</code>	
<code>longling.lib.path.file_exist(filepath)</code>	

<code>longling.lib.candylib.as_list(obj)</code>	A utility function that converts the argument to a list if it is not already.
---	---

<code>longling.lib.clock.print_time(tips[, logger])</code>	
<code>longling.lib.clock.Clock(store_dict, ...[, wall_time process_time tips])</code>	
<code>longling.lib.stream.flush_print(*values, ...)</code>	

`longling.lib.concurrency.
concurrent_pool(...)`

Simple api for start completely independent concurrent programs:

`longling.lib.testing.
simulate_stdin(*inputs)`

.. autosummary:

```
longling.lib.structure.AttrDict
longling.lib.structure.nested_update
longling.lib.structure.SortedList
```

.. autosummary:

```
longling.lib.structure.variable_replace
longling.lib.structure.default_variable_replace
```

3.2.2 candylib

`longling.lib.candylib.as_list(obj) → list`

A utility function that converts the argument to a list if it is not already.

obj (*object*) – argument to be converted to a list

list_obj – If *obj* is a list or tuple, return it. Otherwise, return *[obj]* as a single-element list.

list

```
>>> as_list(1)
[1]
>>> as_list([1])
[1]
>>> as_list((1, 2))
[1, 2]
```

`longling.lib.candylib.dict2pv(dict_obj: dict, path_to_node: list = None)`

```
>>> dict_obj = {"a": {"b": [1, 2], "c": "d"}, "e": 1}
>>> path, value = dict2pv(dict_obj)
>>> path
[['a', 'b'], ['a', 'c'], ['e']]
>>> value
[[1, 2], 'd', 1]
```

`longling.lib.candylib.list2dict(list_obj, value=None, dict_obj=None)`

```
>>> list_obj = ["a", 2, "c"]
>>> list2dict(list_obj, 10)
{'a': {2: {'c': 10}}}
```

`longling.lib.candylib.get_dict_by_path(dict_obj, path_to_node)`

```
>>> dict_obj = {"a": {"b": {"c": 1}}}
>>> get_dict_by_path(dict_obj, ["a", "b", "c"])
1
```

`longling.lib.candylib.format_byte_sizeof(num, suffix='B')`

```
>>> format_byte_sizeof(1024)
'1.00KB'
```

`longling.lib.candylib.group_by_n(obj: list, n: int) → list`

```
>>> list_obj = [1, 2, 3, 4, 5, 6]
>>> group_by_n(list_obj, 3)
[[1, 2, 3], [4, 5, 6]]
```

`longling.lib.candylib.as_ordered_dict(dict_data: (<class 'dict'>, <class 'collections.OrderedDict'>), index: (<class 'list'>, None) = None)`

```
>>> as_ordered_dict({0: 0, 2: 123, 1: 1})
OrderedDict([(0, 0), (2, 123), (1, 1)])
>>> as_ordered_dict({0: 0, 2: 123, 1: 1}, [2, 0, 1])
OrderedDict([(2, 123), (0, 0), (1, 1)])
>>> as_ordered_dict(OrderedDict([(2, 123), (0, 0), (1, 1)]))
OrderedDict([(2, 123), (0, 0), (1, 1)])
```

3.2.3 clock

class `longling.lib.clock.Clock` (*store_dict*: (<class 'dict'>, None) = None, *logger*: (<class 'logging.Logger'>, None) = <Logger clock (INFO)>, *tips*=")

`wall_time` `process_time`

- `wall_time`:
- `process_time`:
- **store_dict** (*dict* or None) – with closure
- **logger** (*logging.logger*) –
- **tips** (*str*) –

```
with Clock():
    a = 1 + 1
clock = Clock()
clock.start()
# some code
clock.end(wall=True) # default to return the wall_time, to get process_time, set_
↪ wall=False
```

end(wall=True)

process_time

start()

wall_time

longling.lib.clock.**print_time**(tips: str = "", logger=<Logger clock (INFO)>)

- **tips**(str) –
- **logger**(logging.Logger or logging) –

```
>>> with print_time("tips"):
...     a = 1 + 1 # The code you want to test
```

longling.lib.clock.**Timer**

longling.lib.clock.Clock

3.2.4 concurrency

longling.lib.concurrency.**concurrent_pool**(level: str, pool_size: int = None, ret: list = None)

Simple api for start completely independent concurrent programs:

- thread
- process
- coroutine

```
def pseudo(idx):
    return idx
```

```
ret = []
with concurrent_pool("p", ret=ret) as e: # or concurrent_pool("t", ret=ret)
    for i in range(4):
        e.submit(pseudo, i)
print(ret)
```

[0, 1, 2, 3]


```
class longling.lib.concurrency.ThreadPool (max_workers=None, thread_name_prefix="",  
                                           ret: list = None)
```

```
submit (fn, *args, **kwargs)
```

Submits a callable to be executed with the given arguments.

Schedules the callable to be executed as `fn(*args, **kwargs)` and returns a Future instance representing the execution of the callable.

A Future representing the given call.

```
class longling.lib.concurrency.ProcessPool (processes=None, *args, ret: list = None,  
                                           **kwargs)
```

3.2.5 formatter

```
longling.lib.formatter.dict_format (data: dict, digits=6, col: int = None)
```

```
>>> print(dict_format({"a": 123, "b": 3, "c": 4, "d": 5})) # doctest: +NORMALIZE_
↪WHITESPACE
a: 123      b: 3      c: 4      d: 5
>>> print(dict_format({"a": 123, "b": 3, "c": 4, "d": 5}, col=3)) # doctest:
↪+NORMALIZE_WHITESPACE
a: 123      b: 3      c: 4
d: 5
```

```
longling.lib.formatter.pandas_format (data: (<class 'dict'>, <class 'list'>, <class 'tuple'>),  
                                         columns: list = None, index: (<class 'list'>, <class  
                                         'str'>) = None, orient='index', pd_kwargs: dict =  
                                         None, max_rows=80, max_columns=80, **kwargs)
```

- **data** (*dict, list, tuple, pd.DataFrame*) –
- **columns** (*list, default None*) – Column labels to use when `orient='index'`. Raises a `ValueError` if used with `orient='columns'`.
- **index** (*list of strings*) – Optional display names matching the labels (same order).
- **orient** (*{'columns', 'index'}, default 'columns'*) – The "orientation" of the data. If the keys of the passed dict should be the columns of the resulting DataFrame, pass 'columns' (default). Otherwise if the keys should be rows, pass 'index'.
- **pd_kwargs** (*dict*) –
- **max_rows** (*(int, None), default 80*) –
- **max_columns** (*(int, None), default 80*) –

```
>>> print(pandas_format({"a": {"x": 1, "y": 2}, "b": {"x": 1.0, "y": 3}}, ["x",
↪"y"]))
      x  y
a  1.0  2
```

0

```

b 1.0 3
>>> print(pandas_format([[1.0, 2], [1.0, 3]], ["x", "y"], index=["a", "b"]))
      x  y
a  1.0  2
b  1.0  3

```

`longling.lib.formatter.table_format` (*data*: (*<class 'dict'>*, *<class 'list'>*, *<class 'tuple'>*), *columns*: *list = None*, *index*: (*<class 'list'>*, *<class 'str'>*) = *None*, *orient*=*'index'*, *pd_kwargs*: *dict = None*, *max_rows*=*80*, *max_columns*=*80*, ***kwargs*)

- **data** (*dict*, *list*, *tuple*, *pd.DataFrame*) –
- **columns** (*list*, *default None*) – Column labels to use when *orient*='index'. Raises a *ValueError* if used with *orient*='columns'.
- **index** (*list of strings*) – Optional display names matching the labels (same order).
- **orient** (*{'columns', 'index'}*, *default 'columns'*) – The "orientation" of the data. If the keys of the passed dict should be the columns of the resulting DataFrame, pass 'columns' (default). Otherwise if the keys should be rows, pass 'index'.
- **pd_kwargs** (*dict*) –
- **max_rows** (*(int, None)*, *default 80*) –
- **max_columns** (*(int, None)*, *default 80*) –

```

>>> print(pandas_format({"a": {"x": 1, "y": 2}, "b": {"x": 1.0, "y": 3}}, ["x",
↪ "y"]))
      x  y
a  1.0  2
b  1.0  3
>>> print(pandas_format([[1.0, 2], [1.0, 3]], ["x", "y"], index=["a", "b"]))
      x  y
a  1.0  2
b  1.0  3

```

`longling.lib.formatter.series_format` (*data*: *dict*, *digits*=*6*, *col*: *int = None*)

```

>>> print(dict_format({"a": 123, "b": 3, "c": 4, "d": 5})) # doctest: +NORMALIZE_
↪ WHITESPACE
a: 123      b: 3      c: 4      d: 5
>>> print(dict_format({"a": 123, "b": 3, "c": 4, "d": 5}, col=3)) # doctest:
↪ +NORMALIZE_WHITESPACE
a: 123      b: 3      c: 4
d: 5

```

3.2.6 iterator

class longling.lib.iterator.**BaseIter**(*src, length=None, *args, **kwargs*)

Notes

- `src`
- `src, reset()`
- `src __length__ BaseIter len()`

```
#
with open("demo.txt") as f:
    bi = BaseIter(f)
    for line in bi:
        pass

#
def open_file():
    with open("demo.txt") as f:
        for line in f:
            yield line

bi = BaseIter(open_file)
for _ in range(5):
    for line in bi:
        pass
    bi.reset()

#
@BaseIter.wrap
def open_file():
    with open("demo.txt") as f:
        for line in f:
            yield line

bi = open_file()
for _ in range(5):
    for line in bi:
        pass
    bi.reset()
```

class longling.lib.iterator.**MemoryIter**(*src, length=None, prefetch=False, *args, **kwargs*)

class longling.lib.iterator.**LoopIter**(*src, length=None, *args, **kwargs*)
`reset()`

class longling.lib.iterator.**AsyncLoopIter**(*src, tank_size=8, timeout=None, level='t'*)
`,`
`reset()`

class longling.lib.iterator.**AsyncIter**(*src, tank_size=8, timeout=None, level='t'*)
`reset()`

```
class longling.lib.iterator.CacheAsyncLoopIter(src, cache_file, rerun=True,  
                                              tank_size=8, timeout=None, level='t')  
    reset(), src function  
  
longling.lib.iterator.iterwrap(itertype: str = 'AsyncLoopIter', *args, **kwargs)  
    AsyncLoopIter
```

```
@iterwrap()  
def open_file():  
    with open("demo.txt") as f:  
        for line in f:  
            yield line  
  
data = open_file()  
for _ in range(5):  
    for line in data:  
        pass
```

3.2.7 loading

```
longling.lib.loading.csv2jsonl(src: ((<class 'str'>, <class 'pathlib.PurePath'>), (<class  
'_io.TextIOWrapper'>, <class 'typing.TextIO'>, <class  
'typing.BinaryIO'>, <class 'codecs.StreamReaderWriter'>,  
<class 'fileinput.FileInput'>)), tar: ((<class 'str'>, <class  
'pathlib.PurePath'>), (<class '_io.TextIOWrapper'>,  
<class 'typing.TextIO'>, <class 'typing.BinaryIO'>, <class  
'codecs.StreamReaderWriter'>, <class 'fileinput.FileInput'>))  
= None, delimiter='', **kwargs)
```

csv /io json /io

transfer csv file or io stream into json file or io stream

- **src** (*PATH_IO_TYPE*) – IO the path to source file or io stream.
- **tar** (*PATH_IO_TYPE*) – IO the path to target file or io stream.
- **delimiter** (*str*) – the delimiter used in csv. some usually used delimiters are "," and "
"
- **kwargs** (*dict*) – options passed to csv.DictWriter

Assume such component is written in demo.csv:

use following codes to reading the component

```
csv2json("demo.csv", "demo.jsonl")
```

and get

```
longling.lib.loading.jsonl2csv(src: ((<class 'str'>, <class 'pathlib.PurePath'>), (<class
'_io.TextIOWrapper'>, <class 'typing.TextIO'>, <class 'typing.BinaryIO'>, <class 'codecs.StreamReaderWriter'>,
<class 'fileinput.FileInput'>)), tar: ((<class 'str'>, <class
'pathlib.PurePath'>), (<class '_io.TextIOWrapper'>,
<class 'typing.TextIO'>, <class 'typing.BinaryIO'>, <class
'codecs.StreamReaderWriter'>, <class 'fileinput.FileInput'>)))
= None, delimiter=', ', **kwargs)
```

json /io csv /io

transfer json file or io stream into csv file or io stream

- **src** (*PATH_IO_TYPE*) – IO the path to source file or io stream.
- **tar** (*PATH_IO_TYPE*) – IO the path to target file or io stream.
- **delimiter** (*str*) – the delimiter used in csv. some usually used delimiters are "," and "
- **kwargs** (*dict*) – options passed to csv.DictWriter

Assume such component is written in demo.csv:

use following codes to reading the component

```
jsonl2csv("demo.csv", "demo.jsonl")
```

and get

```
longling.lib.loading.loading(src: (((<class 'str'>, <class 'pathlib.PurePath'>), (<class
'_io.TextIOWrapper'>, <class 'typing.TextIO'>, <class 'typ-
ing.BinaryIO'>, <class 'codecs.StreamReaderWriter'>, <class
'fileinput.FileInput'>)), Ellipsis), src_type=None)
```

Support read from

- jsonl (apply load_jsonl)
- csv (apply load_csv).
- file in other format will be treated as raw text (apply load_file).
- function will be invoked and return
- others will be directly returned

```
longling.lib.loading.load_jsonl(src: ((<class 'str'>, <class 'pathlib.PurePath'>), (<class
'_io.TextIOWrapper'>, <class 'typing.TextIO'>, <class
'typing.BinaryIO'>, <class 'codecs.StreamReaderWriter'>,
<class 'fileinput.FileInput'>)))
```

jsonl

Assume such component is written in demo.jsonl:

```
for line in load_jsonl('demo.jsonl'):
    print(line)
```

```
longling.lib.loading.load_csv(src: ((<class 'str'>, <class 'pathlib.PurePath'>), (<class
'_io.TextIOWrapper'>, <class 'typing.TextIO'>, <class 'typ-
ing.BinaryIO'>, <class 'codecs.StreamReaderWriter'>, <class
'fileinput.FileInput'>)), delimiter=',', **kwargs)
```

read the dict from csv

Assume such component is written in demo.csv:

```
for line in load_csv('demo.csv'):
    print(line)
```

```
longling.lib.loading.load_file(src: ((<class 'str'>, <class 'pathlib.PurePath'>), (<class
'_io.TextIOWrapper'>, <class 'typing.TextIO'>, <class 'typ-
ing.BinaryIO'>, <class 'codecs.StreamReaderWriter'>, <class
'fileinput.FileInput'>))))
```

Read raw text from source

Assume such component is written in demo.txt:

use following codes to reading the component

```
for line in load_csv('demo.txt'):
    print(line, end="")
```

and get

3.2.8 parser

```
longling.lib.parser.get_class_var(class_obj, exclude_names: (<class 'set'>, None) = None,
get_vars=None) → dict
```

Update in v1.3.18

```
>>> class A(object):
...     att1 = 1
...     att2 = 2
>>> get_class_var(A)
{'att1': 1, 'att2': 2}
>>> get_class_var(A, exclude_names={"att1"})
{'att2': 2}
>>> class B(object):
...     att3 = 3
...     att4 = 4
...     @staticmethod
```

0

()

```
...     def excluded_names():
...         return {"att4"}
>>> get_class_var(B)
{'att3': 3}
```

- **class_obj** –
- **exclude_names** – excluded_names
- **get_vars** –

class_var

```
longling.lib.parser.get_parsable_var(class_obj, parse_exclude: set = None,
                                     dump_parse_functions=None, get_vars=True)
dump_parse_functionsdump
```

```
longling.lib.parser.load_configuration(fp, file_format='json', load_parse_function=None)
Updated in version 1.3.16
```

- **fp** –
- **file_format** –
- **load_parse_function** –

```
longling.lib.parser.var2exp(var_str, env_wrap=<function <lambda>>())
$
```

- **var_str** –
- **env_wrap** –

```
>>> root = "dir"
>>> dataset = "d1"
>>> eval(var2exp("$root/data/$dataset"))
'dir/data/d1'
```

```
longling.lib.parser.path_append(path, *addition, to_str=False)
```

```
path_append("../", "../data", "../dataset1/", "train", to_str=True) '../data/./dataset1/train'
```

- **path** (str or PurePath) –
- **addition** (list(str or PurePath)) –

- **to_str** (*bool*) – Convert the new path to str

```
class longling.lib.parser.Configuration (logger=<module 'logging' from
'/home/docs/.pyenv/versions/3.7.9/lib/python3.7/logging/__init__.py'>,
**kwargs)
```

```
>>> c = Configuration(a=1, b="example", c=[0,2], d={"a1": 3})
>>> c.instance_var
{'a': 1, 'b': 'example', 'c': [0, 2], 'd': {'a1': 3}}
>>> c.default_file_format()
'json'
>>> c.get("a")
1
>>> c.get("e") is None
True
>>> c.get("e", 0)
0
>>> c.update(e=2)
>>> c["e"]
2
```

class_var

parameters – all variables used as parameters

dict

dump (*cfg_path: str, override=True, file_format=None*)

Updated in version 1.3.16

- **cfg_path** (*str*) –

- **override** (*bool*) –

- **file_format** (*str*) –

classmethod excluded_names ()

exclude names set –

set

classmethod load (*cfg_path, file_format=None, **kwargs*)

Updated in version 1.3.16

classmethod load_cfg (*cfg_path, file_format=None, **kwargs*)

parsable_var

store_vars –

dict

```
class longling.lib.parser.ConfigurationParser (class_type, excluded_names: (<class
'set'>, None) = None, commands=None,
*args, params_help=None, com-
mands_help=None, override_help=False,
**kwargs)
```

Update in v1.3.18

click_obj "-att_name att_value" '-kwargs'


```
--kwargs key1=value1;key2=value2;...
```

```
cli_parser = ConfigurationParser(Configuration)
```

```
cli_parser = ConfigurationParser($function)
```

```
cli_parser = ConfigurationParser([$function1, $function2])
```

- `cli_parser()`
- `cli_parser('$parameter1 $parameters ...')`
- `cli_parser(["--a", "int(1)", "--b", "int(2)"])`

Notes

int, float, dict, list, set, tuple, None

- **class_type** –
- **excluded_names** –
- **commands** –

```
>>> class TestC(Configuration):
...     a = 1
...     b = 2
>>> def test_f1(k=1):
...     return k
>>> def test_f2(h=1):
...     return h
>>> def test_f3(m):
...     return m
>>> parser = ConfigurationParser(TestC)
>>> parser("--a 1 --b 2")
{'a': '1', 'b': '2'}
>>> ConfigurationParser.get_cli_cfg(TestC)
{'a': 1, 'b': 2}
>>> parser(["--a", "1", "--b", "int(1)"])
{'a': '1', 'b': 1}
>>> parser(["--a", "1", "--b", "int(1)", "--kwargs", "c=int(3);d=None"])
{'a': '1', 'b': 1, 'c': 3, 'd': None}
>>> parser.add_command(test_f1, test_f2, test_f3)
>>> parser(["test_f1"])
{'a': 1, 'b': 2, 'k': 1, 'subcommand': 'test_f1'}
>>> parser(["test_f2"])
{'a': 1, 'b': 2, 'h': 1, 'subcommand': 'test_f2'}
>>> parser(["test_f3", "3"])
{'a': 1, 'b': 2, 'm': 3, 'subcommand': 'test_f3'}
```

0

()

```

>>> parser = ConfigurationParser(TestC, commands=[test_f1, test_f2])
>>> parser(["test_f1"])
{'a': 1, 'b': 2, 'k': 1, 'subcommand': 'test_f1'}
>>> class TestCC:
...     c = {"_c": 1, "_d": 0.1}
>>> parser = ConfigurationParser(TestCC)
>>> parser("--c _c=int(3);_d=float(0.3)")
{'c': {'_c': 3, '_d': 0.3}}
>>> class TestCls:
...     def a(self, a=1):
...         return a
...     @staticmethod
...     def b(b=2):
...         return b
...     @classmethod
...     def c(cls, c=3):
...         return c
>>> parser = ConfigurationParser(TestCls, commands=[TestCls.b, TestCls.c])
>>> parser("b")
{'b': 2, 'subcommand': 'b'}
>>> parser("c")
{'c': 3, 'subcommand': 'c'}

```

add_command (*commands, help_info: (typing.List[typing.Dict], <class 'dict'>, <class 'str'>, None) = None)

static func_spec (f)

classmethod get_cli_cfg (params_class) → dict

static parse (arguments)

class longling.lib.parser.**Formatter** (formatter: (<class 'str'>, None) = None)

```

>>> formatter = Formatter()
>>> formatter("hello world")
'hello world'
>>> formatter = Formatter("hello {}")
>>> formatter("world")
'hello world'
>>> formatter = Formatter("hello {} v{:.2f}")
>>> formatter("world", 0.2)
'hello world v0.20'
>>> formatter = Formatter("hello {1} v{0:.2f}")
>>> formatter(0.2, "world")
'hello world v0.20'
>>> Formatter.format(0.2, "world", formatter="hello {1} v{0:.3f}")
'hello world v0.200'

```

class longling.lib.parser.**ParserGroup** (parsers: dict, prog=None, usage=None, description=None, epilog=None, add_help=True)

```

>>> class TestC(Configuration):
...     a = 1
...     b = 2
>>> def test_f1(k=1):

```

()

()

```

...     return k
>>> def test_f2(h=1):
...     return h
>>> class TestC2(Configuration):
...     c = 3
>>> parser1 = ConfigurationParser(TestC, commands=[test_f1])
>>> parser2 = ConfigurationParser(TestC, commands=[test_f2])
>>> pg = ParserGroup({"model1": parser1, "model2": parser2})
>>> pg(["model1", "test_f1"])
{'a': 1, 'b': 2, 'k': 1, 'subcommand': 'test_f1'}
>>> pg(["model2 test_f2"])
{'a': 1, 'b': 2, 'h': 1, 'subcommand': 'test_f2'}

```

longling.lib.parser.is_classmethod(*method*)

method –

```

>>> class A:
...     def a(self):
...         pass
...     @staticmethod
...     def b():
...         pass
...     @classmethod
...     def c(cls):
...         pass
>>> obj = A()
>>> is_classmethod(obj.a)
False
>>> is_classmethod(obj.b)
False
>>> is_classmethod(obj.c)
True
>>> def fun():
...     pass
>>> is_classmethod(fun)
False

```

3.2.9 path

longling.lib.path.path_append(*path*, **addition*, *to_str=False*)

```
path_append("../", "../data", "../dataset1", "train", to_str=True) '../data/./dataset1/train'
```

- **path** (*str* or *PurePath*) –
- **addition** (*list* (*str* or *PurePath*)) –
- **to_str** (*bool*) – Convert the new path to str

`longling.lib.path.file_exist(filepath)`

`longling.lib.path.abs_current_dir(filepath)`

Example

`longling.lib.path.type_from_name(filename)`

```
>>> type_from_name("1.txt")
'.txt'
```

`longling.lib.path.tmpfile(suffix=None, prefix=None, dir=None)`

Create a temporary file, which will automatically cleaned after used (outside "with" closure).

3.2.10 progress

epochbatch

tqdm progressdescription

- **MonitorPlayer (better than tqdm, where only n is changed and description is fixed)**

- `__call__`

- **ProgressMonitor**

- `ProgressMonitor__call__ IterableMIcing`

- `__init__ MonitorPlayer`

- **IterableMIcing**

```
class DemoMonitor(ProgressMonitor):
    def __call__(self, iterator):
        return IterableMIcing(
            iterator,
            self.player, self.player.set_length
        )

progress_monitor = DemoMonitor(MonitorPlayer())

for _ in range(5):
    for _ in progress_monitor(range(10000)):
        pass
    print()
```

cooperate with tqdm

```
from tqdm import tqdm

class DemoTqdmMonitor(ProgressMonitor):
    def __call__(self, iterator, **kwargs):
        return tqdm(iterator, **kwargs)
```

```

class longling.lib.progress.IterableMIcing (iterator: (typing.Iterable, <class 'list'>,
                                                         <class 'tuple'>, <class 'dict'>),
                                                         hook_in_iter=<function pass_function>,
                                                         hook_after_iter=<function pass_function>,
                                                         length: (<class 'int'>, None) = None)
    * count, count + 1, count * __iter__ call_in_iter * call_after_iter

```

```

• iterator –
• hook_in_iter – , count
• hook_after_iter – length
• length –
• iterator = IterableMIcing(range(100)) (>>>) –
• for i in iterator (>>>) –
• pass (..) –
• len(iterator) (>>>) –
• 100 –
• def iter_fn(num) (>>>) –
• for i in range(num) (..) –
• yield num (..) –
• iterator = IterableMIcing(iter_fn(50)) (>>>) –
• for i in iterator –
• pass –
• len(iterator) –
• 50 –

```

```
class longling.lib.progress.MonitorPlayer
```

```
class longling.lib.progress.AsyncMonitorPlayer (cache_size=10000)
```

3.2.11 regex

```

longling.lib.regex.variable_replace (string: str, key_lower: bool = True, quotation: str = ",
                                     **variables)

```

```

>>> string = "hello $who"
>>> variable_replace(string, who="world")
'hello world'
>>> string = "hello $WHO"
>>> variable_replace(string, key_lower=False, WHO="world")
'hello world'
>>> string = "hello $WHO"
>>> variable_replace(string, who="longling")

```

()

()

```
'hello longling'
>>> string = "hello $Wh_o"
>>> variable_replace(string, wh_o="longling")
'hello longling'
```

`longling.lib.regex.default_variable_replace` (*string: str, default_value: (<class 'str'>, None, <class 'dict'>) = None, key_lower: bool = True, quotation: str = ", **variables)*
→ str

```
>>> string = "hello $who, I am $author"
>>> default_variable_replace(string, default_value={"author": "groot", who="world"},
↪)
'hello world, I am groot'
>>> string = "hello $who, I am $author"
>>> default_variable_replace(string, default_value={"author": "groot"})
'hello , I am groot'
>>> string = "hello $who, I am $author"
>>> default_variable_replace(string, default_value=' ', who="world")
'hello world, I am '
>>> string = "hello $who, I am $author"
>>> default_variable_replace(string, default_value=None, who="world")
'hello world, I am $author'
```

3.2.12 stream

`longling.lib.stream.to_io` (*stream: (<class '_io.TextIOWrapper'>, <class 'typing.TextIO'>, <class 'typing.BinaryIO'>, (<class 'str'>, <class 'pathlib.PurePath'>), <class 'list'>, None) = None, mode='r', encoding='utf-8', **kwargs)*

Convert an object as an io stream, could be a path to file or an io stream.

```
to_io("demo.txt") # equal to open("demo.txt")
to_io(open("demo.txt")) # equal to open("demo.txt")
a = to_io() # equal to a = sys.stdin
b = to_io(mode="w") # equal to a = sys.stdout
```

`longling.lib.stream.as_io` (*src: (<class '_io.TextIOWrapper'>, <class 'typing.TextIO'>, <class 'typing.BinaryIO'>, (<class 'str'>, <class 'pathlib.PurePath'>), <class 'list'>, None) = None, mode='r', encoding='utf-8', **kwargs)*

with wrapper for `to_io` function, default mode is "r"

```
with as_io("demo.txt") as f:
    for line in f:
        pass
```

()

()

```

# equal to
with open(demo.txt) as src:
    with as_io(src) as f:
        for line in f:
            pass

# from several files
with as_io(["demo1.txt", "demo2.txt"]) as f:
    for line in f:
        pass

# from sys.stdin
with as_io() as f:
    for line in f:
        pass

```

longling.lib.stream.**as_out_io**(tar: (<class 'io.TextIOWrapper'>, <class 'typing.TextIO'>, <class 'typing.BinaryIO'>, (<class 'str'>, <class 'pathlib.PurePath'>)), <class 'list'>, None) = None, mode='w', encoding='utf-8', **kwargs)
 with wrapper for to_io function, default mode is "w"

```

with as_out_io("demo.txt") as wf:
    print("hello world", file=wf)

# equal to
with open(demo.txt) as tar:
    with as_out_io(tar) as f:
        print("hello world", file=wf)

# to sys.stdout
with as_out_io() as wf:
    print("hello world", file=wf)

# to sys.stderr
with as_out_io(mode="stderr") as wf:
    print("hello world", file=wf)

```

longling.lib.stream.**wf_open**(stream_name: (((<class 'str'>, <class 'pathlib.PurePath'>), (<class 'io.TextIOWrapper'>, <class 'typing.TextIO'>, <class 'typing.BinaryIO'>, <class 'codecs.StreamReaderWriter'>, <class 'fileinput.FileInput'>)), None) = None, mode='w', encoding='utf-8', **kwargs)

Simple wrapper to codecs for writing.

stream_name mode - w stderr; stdout

stream_name

- **stream_name**(str, PurePath or None) –
- **mode**(str) –

- **encoding**(*str*) – utf-8
- write_stream** –
StreamReaderWriter

```
>>> wf = wf_open(mode="stdout")
>>> print("hello world", file=wf)
hello world
```

`longling.lib.stream.close_io`(*stream*)
sys.stdin, sys.stdout, sys.stderr

`longling.lib.stream.flush_print`(**values, **kwargs*)

`longling.lib.stream.build_dir`(*path, mode=509, parse_dir=True*)
path

- **path**(*str*) –
- **mode**(*int*) –
- **parse_dir**(*bool*) –

class `longling.lib.stream.AddPrinter`(*fp, values_wrapper=<function dPrinter.<lambda>>, to_io_params=None, ensure_io=False, **kwargs*)

add

```
>>> import sys
>>> printer = AddPrinter(sys.stdout, ensure_io=True)
>>> printer.add("hello world")
hello world
```

exception `longling.lib.stream.StreamError`

`longling.lib.stream.check_file`(*filepath, size=None*)
size

- **filepath**(*str*) –
 - **size**(*int*) –
- file exist or not**
bool

`longling.lib.stream.encode`(*src, src_encoding, tar, tar_encoding*)
Convert a file in source encoding to target encoding

- **src** –

- **src_encoding** –
- **tar** –
- **tar_encoding** –

`longling.lib.stream.block_std()`

```
>>> print("hello world")
hello world
>>> with block_std():
...     print("hello world")
```

3.2.13 structure

class `longling.lib.structure.AttrDict(*args, **kwargs)`

Example

```
>>> ad = AttrDict({'first_name': 'Eduardo'}, last_name='Pool', age=24, sports=[
↪ 'Soccer'])
>>> ad
{'first_name': 'Eduardo', 'last_name': 'Pool', 'age': 24, 'sports': ['Soccer']}
>>> ad.first_name
'Eduardo'
>>> ad.age
24
>>> ad.age = 16
>>> ad.age
16
>>> ad["age"] = 20
>>> ad["age"]
20
```

class `longling.lib.structure.SortedList(iterable: Iterable[T_co] = (), key=None)`

A list maintaining the element in an ascending order.

A custom key function can be supplied to customize the sort order.

```
>>> sl = SortedList()
>>> sl.adds(*[1, 2, 3, 4, 5])
>>> sl
[1, 2, 3, 4, 5]
>>> sl.add(7)
>>> sl
[1, 2, 3, 4, 5, 7]
>>> sl.add(6)
>>> sl
[1, 2, 3, 4, 5, 6, 7]
```

0

```

>>> sl = SortedList([4])
>>> sl.add(3)
>>> sl.add(2)
>>> sl
[2, 3, 4]
>>> list(reversed(sl))
[4, 3, 2]
>>> sl = SortedList([("harry", 1), ("tom", 0)], key=lambda x: x[1])
>>> sl
[('tom', 0), ('harry', 1)]
>>> sl.add(("jack", -1), key=lambda x: x[1])
>>> sl
[('jack', -1), ('tom', 0), ('harry', 1)]
>>> sl.add(("ada", 2))
>>> sl
[('jack', -1), ('tom', 0), ('harry', 1), ('ada', 2)]

```

`longling.lib.structure.nested_update(src: dict, update: dict)`

```

>>> nested_update({"a": {"x": 1}}, {"a": {"y": 2}})
{'a': {'x': 1, 'y': 2}}
>>> nested_update({"a": {"x": 1}}, {"a": {"x": 2}})
{'a': {'x': 2}}
>>> nested_update({"a": {"x": 1}}, {"b": {"y": 2}})
{'a': {'x': 1}, 'b': {'y': 2}}
>>> nested_update({"a": {"x": 1}}, {"a": 2})
{'a': 2}

```

3.2.14 testing

`longling.lib.testing.simulate_stdin(*inputs)`
`inputs` (list of str)–

```

>>> with simulate_stdin("12", "", "34"):
...     a = input()
...     b = input()
...     c = input()
>>> a
'12'
>>> b
''
>>> c
'34'

```

3.2.15 time

`longling.lib.time.get_current_timestamp() → str`

```
> get_current_timestamp()
'20200327172235'
```

3.2.16 utilog

`longling.lib.utilog.config_logging` (*filename=None, log_format=None, level=20, logger=None, console_log_level=None, propagate=False, mode='a', file_format=None, encoding: (<class 'str'>, None) = 'utf-8', enable_colored=False, datefmt=None*)

- **filename** (*str or None*) –
- **log_format** (*str*) – : *%(name)s, %(levelname)s %(message)s datefmt %(name)s: %(asctime)s, %(levelname)s %(message)s*
- **level** (*str or int*) –
- **logger** (*str or logging.logger*) – loggerroot logger loggerlogger
- **console_log_level** (*str, int or None*) –
- **propagate** (*bool*) –
- **mode** (*str*) –
- **file_format** (*str or None*) – log_format
- **encoding** –
- **enable_colored** (*bool*) –
- **datefmt** (*str*) –

`longling.lib.utilog.default_timestamp()` → *str*

```
> get_current_timestamp()
'20200327172235'
```

3.2.17 yaml

class `longling.lib.yaml_helper.FoldedString`

`longling.lib.yaml_helper.dump_folded_yaml` (*yaml_string*)
 specially designed for arch module, should not be used in other places

`longling.lib.yaml_helper.ordered_yaml_load` (*stream, Loader=<class 'yaml.loader.Loader'>, object_pairs_hook=<class 'collections.OrderedDict'>*)

```
ordered_yaml_load("path_to_file.yaml")
OrderedDict({"a":123})
```

3.3 Spider

1. url
 - [x]
2.
 - [x] url
 - [x]
 - []

3.3.1 Quick Glance

<code>longling.spider.lib.get_html_code(url)</code>	get encoded html code from specified url
<code>longling.spider.download_data.download_file(url)</code>	cli alias: download, download data from specified url

`longling.spider.lib.get_html.get_html_code(url)`
get encoded html code from specified url

`longling.spider.download_data.download_file(url, save_path=None, override=True, decomp=True, reporthook=None)`
cli alias: download, download data from specified url

- **url** –
- **save_path** –
- **override** –
- **decomp** –
- **reporthook** –

3.4 Architecture Tools for Constructing Projects

3.4.1 notice

- In sphinx setting, the default setting in sphinx-quickstart would be

```
> Separate source and build directories (y/n) [n]
```

Thus, default directory to the built files is `_build`. If the `y` is chosen, the directory to the built files is `build`.

3.4.2 entrance

```
longling.Architecture.cli.main.cli (skip_top=True,      project=None,      override=None,
                                     tar_dir='./', **kwargs)
```

The main function for arch

3.4.3 components

```
longling.Architecture.install_file.template_copy (src: (<class 'str'>, <class 'path-
lib.PurePath'>), tar: (<class 'str'>,
                      <class 'pathlib.PurePath'>), de-
                      fault_value: (<class 'str'>, <class
                      'dict'>, None) = "", quotation="'",
                      key_lower=True, **variables)
```

Generate the tar file based on the template file where the variables will be replaced. Usually, the variable is specified like `$PROJECT` in the template file.

- **src** (template file) –
- **tar** (target location) –
- **default_value** (the default value) –
- **quotation** (the quotation to wrap the variable value) –
- **variables** (the real variable values which are used to replace the variable in template file) –

```
longling.Architecture.install_file.gitignore (atype: str = "", tar_dir: (<class 'str'>,
                                                                           <class 'pathlib.PurePath'>) = './')
cli alias: arch gitignore
```

- **atype** (the gitignore type, currently support *docs* and *python*) –
- **tar_dir** (target directory) –

```
longling.Architecture.install_file.pytest (tar_dir: (<class 'str'>, <class 'path-
lib.PurePath'>) = './')
cli alias: arch pytest
```

tar_dir –

```
longling.Architecture.install_file.coverage (tar_dir: (<class 'str'>, <class 'path-
lib.PurePath'>) = './', **variables)
cli alias: arch coverage
```

- **tar_dir** –
- **variables** – These variables should be provided:
 - project

```
longling.Architecture.install_file.pysetup (tar_dir='./', **variables)
cli alias: arch pysetup
```

- **tar_dir** –

- **variables** –

```
longling.Architecture.install_file.sphinx_conf (tar_dir='./', **variables)
cli alias: arch sphinx_conf
```

- **tar_dir** –

- **variables** –

```
longling.Architecture.install_file.makefile (tar_dir='./', **variables)
cli alias: arch makefile
```

- **tar_dir** –

- **variables** –

```
longling.Architecture.install_file.readthedocs (tar_dir='./')
cli alias: arch readthedocs
```

- **tar_dir** –

```
longling.Architecture.install_file.travis (tar_dir: (<class 'str'>, <class 'pathlib.PurePath'>) = './')
cli alias: arch travis
```

- **tar_dir** –

```
longling.Architecture.install_file.nni (tar_dir='./')
cli alias: arch nni and install nni
```

- **tar_dir** –

```
longling.Architecture.install_file.dockerfile (atype, tar_dir='./', **variables)
cli alias: arch dockerfile
```

- **atype** –

- **tar_dir** –

- **variables** –

```
longling.Architecture.install_file.gitlab_ci (private, stages: dict, atype: str = "", tar_dir:
(<class 'str'>, <class 'pathlib.PurePath'>)
= './', version_in_path=True)
```

```
cli alias: arch gitlab_ci
```

- **private** –

- **stages** –

- **atype** –

- **tar_dir** –

- **version_in_path** –

```
longling.Architecture.install_file.chart (tar_dir: (<class 'str'>, <class 'pathlib.PurePath'>) = './')
cli alias: arch chart
```

`tar_dir(target directory)-`

`longling.Architecture.utils.legal_input` (`__prompt: str`, `__legal_input: set = None`, `__illegal_input: set = None`, `is_legal=None`, `__default_value: str = None`)

To make sure the input legal, if the input is illegal, the user will be asked to retype the input.

Input being illegal means the input either not in `__legal_input` or in `__illegal_input`.

For the case that user do not type in anything, if the `__default_value` is set, the `__default_value` will be used. Else, the input will be treated as illegal.

- `__prompt` (`str`) – tips to be displayed
- `__legal_input` (`set`) – the input should be in the `__legal_input` set
- `__illegal_input` (`set`) – the input should be not in the `__illegal_input` set
- `is_legal` (`function`) – the function used to judge whether the input is legal, by default, we use the inner `__is_legal` function
- `__default_value` (`str`) – default value when user type in nothing

3.5 Machine Learning

3.5.1 ML Framework

ML Framework is designed to help quickly construct a practical ML program where the user can focus on developing the algorithm despite of some other additional but important engineering components like `log` and `cli`.

Currently, two supported packages are provided for the popular DL framework: `mxnet` and `pytorch`. The overall scenery are almost the same, but the details may be a little different.

To be noticed that, ML Framework just provide a template, all components are allowed to be modified.

Overview

The architecture produced by the ML Framework is look like:

```
ModelName/
├── __init__.py
├── docs/
├── ModelName/
├── REAME.md
└── Some other components
```

And the core part is `ModelName` under the package `ModelName`, the architecture of it is:

```
ModelName/
├── __init__.py
├── ModelName.py          <-- the main module
├── Module/
│   ├── __init__.py
│   ├── configuration.py  <-- define the configurable variables
│   └── etl.py            <-- define how the data will be loaded and_
└── <pre>processed
```

0

└─ module.py	<-- the wrapper of the network, rarely need
↪modification	
└─ run.py	<-- human testing script
└─ sym/	<-- define the network
└─ __init__.py	
└─ fit_eval.py	<-- define how the network will be trained and
↪evaluated	
└─ net.py	<-- network architecture
└─ viz.py	<-- (option) how to visualize the network

Configuration

In configuration, some variables are predefined, such as the `data_dir`(where the data is stored) and `model_dir`(where the model file like parameters and running log should be stored). The following rules is used to automatically construct the needed path, which can be modified as the user wants:

```
model_name = "automatically be consistent with ModelName"

root = "/"
dataset = "" # option
timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S") # option
workspace = "" # option

root_data_dir = "$root/data/$dataset" if dataset else "$root/data"
data_dir = "$root_data_dir/data"
root_model_dir = "$root_data_dir/model/$model_name"
model_dir = "$root_model_dir/$workspace" if workspace else root_model_dir
cfg_path = "$model_dir/configuration.json"
```

The value of the variable containing \$ will be automatically evaluated during program running. Thus, it is easy to construct flexible variables via cli. For example, some one may want to have the `model_dir` contained timestamp information, then he or she can specify the `model_dir` in cli like:

```
--model_dir \ $root_model_dir/\ $workspace/\ $timestamp
```

Annotation: \ is a escape character in shell, which can have \ \$variable finally be got as the string \$variable in the program, otherwise, the variable will be converted to the environment variable of shell like \$HOME.

Also, some general variables which may frequently used in all algorithms are also predefined like `optimizer` and `batch_size`:

```
#
begin_epoch = 0
end_epoch = 100
batch_size = 32
save_epoch = 1

#
optimizer, optimizer_params = get_optimizer_cfg(name="base")
lr_params = {
    "learning_rate": optimizer_params["learning_rate"],
    "step": 100,
    "max_update_steps": get_update_steps(
        update_epoch=10,
        batches_per_epoch=1000,
```



```

    },
}

```

3.5.2 metrics

Metrics

classification

```

longling.ML.metrics.classification.classification_report(y_true, y_pred=None,
                                                         y_score=None, labels=None, metrics=None, sample_weight=None,
                                                         average_options=None, multi-class_to_multilabel=False,
                                                         logger=<module 'logging' from
                                                         '/home/docs/.pyenv/versions/3.7.9/lib/python3.7/lo
                                                         **kwargs>)

```

Currently support binary and multiclass classification.

- **y_true** (*list, 1d array-like, or label indicator array / sparse matrix*) – Ground truth (correct) target values.
- **y_pred** (*list or None, 1d array-like, or label indicator array / sparse matrix*) – Estimated targets as returned by a classifier.
- **y_score** (*array or None, shape = [n_samples] or [n_samples, n_classes]*) – Target scores, can either be probability estimates of the positive class, confidence values, or non-thresholded measure of decisions (as returned by "decision_function" on some classifiers). For binary y_true, y_score is supposed to be the score of the class with greater label.
- **labels** (*array, shape = [n_labels]*) – Optional list of label indices to include in the report.
- **metrics** (*list of str,*) – Support: precision, recall, f1, support, accuracy, auc, aupoc.
- **sample_weight** (*array-like of shape = [n_samples], optional*) – Sample weights.
- **average_options** (*str or list*) – default to macro, choices (one or many): "micro", "macro", "samples", "weighted"
- **multiclass_to_multilabel** (*bool*) –
- **logger** –

```

>>> import numpy as np
>>> # binary classification
>>> y_true = np.array([0, 0, 1, 1, 0])
>>> y_pred = np.array([0, 1, 0, 1, 0])
>>> classification_report(y_true, y_pred)
              precision    recall  f1   support
0               0.666667    0.666667    0.666667         3
1               0.500000    0.500000    0.500000         2
macro_avg       0.583333    0.583333    0.583333         5
accuracy: 0.600000
>>> y_true = np.array([0, 0, 1, 1])
>>> y_score = np.array([0.1, 0.4, 0.35, 0.8])
>>> classification_report(y_true, y_score=y_score)      # doctest: +NORMALIZE_
↪WHITESPACE
macro_auc: 0.750000 macro_aupoc: 0.833333
>>> y_true = np.array([0, 0, 1, 1])
>>> y_pred = [0, 0, 0, 1]
>>> y_score = np.array([0.1, 0.4, 0.35, 0.8])
>>> classification_report(y_true, y_pred, y_score=y_score)      # doctest: _
↪+NORMALIZE_WHITESPACE
              precision    recall  f1   support
0               0.666667    1.00    0.800000         2
1               1.000000    0.50    0.666667         2
macro_avg       0.833333    0.75    0.733333         4
accuracy: 0.750000 macro_auc: 0.750000 macro_aupoc: 0.833333
>>> # multiclass classification
>>> y_true = [0, 1, 2, 2, 2]
>>> y_pred = [0, 0, 2, 2, 1]
>>> classification_report(y_true, y_pred)
              precision    recall  f1   support
0               0.5    1.000000    0.666667         1
1               0.0    0.000000    0.000000         1
2               1.0    0.666667    0.800000         3
macro_avg       0.5    0.555556    0.488889         5
accuracy: 0.600000
>>> # multiclass in multilabel
>>> y_true = np.array([0, 0, 1, 1, 2, 1])
>>> y_pred = np.array([2, 1, 0, 2, 1, 0])
>>> y_score = np.array([
...     [0.15, 0.4, 0.45],
...     [0.1, 0.9, 0.0],
...     [0.333333, 0.333333, 0.333333],
...     [0.15, 0.4, 0.45],
...     [0.1, 0.9, 0.0],
...     [0.333333, 0.333333, 0.333333]
... ])
>>> classification_report(
...     y_true, y_pred, y_score,
...     multiclass_to_multilabel=True,
...     metrics=["aupoc"]
... )
              aupoc
0               0.291667
1               0.416667
2               0.166667

```

()

()

```

macro_avg 0.291667
>>> classification_report(
...     y_true, y_pred, y_score,
...     multiclass_to_multilabel=True,
...     metrics=["auc", "aupoc"]
... )
          auc      aupoc
0          0.250000  0.291667
1          0.055556  0.416667
2          0.100000  0.166667
macro_avg 0.135185  0.291667
macro_auc: 0.194444
>>> y_true = np.array([0, 1, 1, 1, 2, 1])
>>> y_pred = np.array([2, 1, 0, 2, 1, 0])
>>> y_score = np.array([
...     [0.45, 0.4, 0.15],
...     [0.1, 0.9, 0.0],
...     [0.33333, 0.33333, 0.33333],
...     [0.15, 0.4, 0.45],
...     [0.1, 0.9, 0.0],
...     [0.33333, 0.33333, 0.33333]
... ])
>>> classification_report(
...     y_true, y_pred,
...     y_score,
...     multiclass_to_multilabel=True,
... ) # doctest: +NORMALIZE_WHITESPACE
      precision    recall  f1   auc      aupoc  support
0          0.000000  0.000000  0.000000  1.00  1.000000         1
1          0.500000  0.250000  0.333333  0.25  0.583333         4
2          0.000000  0.000000  0.000000  0.10  0.166667         1
macro_avg 0.166667  0.083333  0.111111  0.45  0.583333         6
accuracy: 0.166667  macro_auc: 0.437500
>>> classification_report(
...     y_true, y_pred,
...     y_score,
...     labels=[0, 1],
...     multiclass_to_multilabel=True,
... ) # doctest: +NORMALIZE_WHITESPACE
      precision    recall  f1   auc      aupoc  support
0          0.00    0.000  0.000000  1.00  1.000000         1
1          0.50    0.250  0.333333  0.25  0.583333         4
macro_avg 0.25    0.125  0.166667  0.45  0.583333         5
accuracy: 0.166667  macro_auc: 0.437500

```

regression

`longling.ML.metrics.regression.regression_report` (*y_true*, *y_pred*, *metrics=None*, *sample_weight=None*, *multioutput='uniform_average'*, *average_options=None*, *key_prefix=""*, *key_suffix=""*, *verbose=True*)

- **y_true** (array-like of shape (*n_samples*,) or (*n_samples*, *n_outputs*)) – Ground truth (correct) target values.

- **y_pred** (array-like of shape (n_samples,) or (n_samples, n_outputs)) – Estimated target values.
- **metrics** (list of str,) – Support: evar(explained_variance), mse, rmse, mae, r2
- **sample_weight** (array-like of shape (n_samples,), optional) – Sample weights.
- **multioutput** (string in ['raw_values', 'uniform_average', 'variance_weighted'], list) – or array-like of shape (n_outputs) Defines aggregating of multiple output values. Disabled when verbose is True. Array-like value defines weights used to average errors. 'raw_values' :

Returns a full set of errors in case of multioutput input.

'uniform_average': Errors of all outputs are averaged with uniform weight. Alias: "macro"

'variance_weighted': Only support in evar and r2. Scores of all outputs are averaged, weighted by the variances of each individual output. Alias: "vw"

- **average_options** (str or list) – default to macro, choices (one or many): "macro", "vw"
- **key_prefix** (str) –
- **key_suffix** (str) –
- **verbose** (bool) –
- **evar** (explained variance)
- **mse** (mean squared error)
- **rmse** (root mean squared error)
- **mae** (mean absolute error)
- **r2** (r2 score)

```
>>> y_true = [[0.5, 1, 1], [-1, 1, 1], [7, -6, 1]]
>>> y_pred = [[0, 2, 1], [-1, 2, 1], [8, -5, 1]]
>>> regression_report(y_true, y_pred)      # doctest: +NORMALIZE_WHITESPACE
           evar      mse      rmse  mae      r2
0           0.967742  0.416667  0.645497  0.5  0.965438
1           1.000000  1.000000  1.000000  1.0  0.908163
2           1.000000  0.000000  0.000000  0.0  1.000000
uniform_average  0.989247  0.472222  0.548499  0.5  0.957867
variance_weighted 0.983051  0.472222  0.548499  0.5  0.938257
>>> regression_report(y_true, y_pred, verbose=False) # doctest: +NORMALIZE_
↪WHITESPACE
evar: 0.989247      mse: 0.472222      rmse: 0.548499      mae: 0.500000      r2: 0.957867
>>> regression_report(
...     y_true, y_pred, multioutput="variance_weighted", verbose=False
... ) # doctest: +NORMALIZE_WHITESPACE
evar: 0.983051      mse: 0.472222      rmse: 0.548499      mae: 0.500000      r2: 0.938257
```

()

```

>>> regression_report(y_true, y_pred, multioutput=[0.3, 0.6, 0.1], verbose=False)
↳ # doctest: +NORMALIZE_WHITESPACE
evar: 0.990323      mse: 0.725000      rmse: 0.793649      mae: 0.750000      r2: 0.934529
>>> regression_report(y_true, y_pred, verbose=True) # doctest: +NORMALIZE_
↳WHITESPACE

```

	evar	mse	rmse	mae	r2
0	0.967742	0.416667	0.645497	0.5	0.965438
1	1.000000	1.000000	1.000000	1.0	0.908163
2	1.000000	0.000000	0.000000	0.0	1.000000
uniform_average	0.989247	0.472222	0.548499	0.5	0.957867
variance_weighted	0.983051	0.472222	0.548499	0.5	0.938257

```

>>> regression_report(
...     y_true, y_pred, verbose=True, average_options=["macro", "vw", [0.3, 0.6,
↳0.1]]
... ) # doctest: +NORMALIZE_WHITESPACE

```

	evar	mse	rmse	mae	r2
0	0.967742	0.416667	0.645497	0.50	0.965438
1	1.000000	1.000000	1.000000	1.00	0.908163
2	1.000000	0.000000	0.000000	0.00	1.000000
uniform_average	0.989247	0.472222	0.548499	0.50	0.957867
variance_weighted	0.983051	0.472222	0.548499	0.50	0.938257
weighted	0.990323	0.725000	0.793649	0.75	0.934529

ranking

longling.ML.metrics.ranking.**ranking_report**(y_true, y_pred, k: (<class 'int'>, <class 'list'>)) = None, continuous=False, coerce='ignore', pad_pred=-100, metrics=None, bottom=False, verbose=True) → longling.ML.metrics.utils.POrderedDict

- **y_true** –
- **y_pred** –
- **k** –
- **continuous** –
- **coerce** –
- **pad_pred** –
- **metrics** –
- **bottom** –
- **verbose** –

```

>>> y_true = [[1, 0, 0], [0, 0, 1]]
>>> y_pred = [[0.75, 0.5, 1], [1, 0.2, 0.1]]
>>> ranking_report(y_true, y_pred) # doctest: +NORMALIZE_WHITESPACE
ndcg@k precision@k recall@k f1@k len@k support@k

```

()

```

1  1.000000  0.000000  0.0  0.0  1.0  2
3  0.565465  0.333333  1.0  0.5  3.0  2
5  0.565465  0.333333  1.0  0.5  3.0  2
10 0.565465  0.333333  1.0  0.5  3.0  2
auc: 0.250000      map: 0.416667  mrr: 0.416667  coverage_error: 2.500000
↳ ranking_loss: 0.750000  len: 3.000000
support: 2
>>> ranking_report(y_true, y_pred, k=[1, 3, 5]) # doctest: +NORMALIZE_WHITESPACE
      ndcg@k  precision@k  recall@k  f1@k  len@k  support@k
1  1.000000  0.000000  0.0  0.0  1.0  2
3  0.565465  0.333333  1.0  0.5  3.0  2
5  0.565465  0.333333  1.0  0.5  3.0  2
auc: 0.250000      map: 0.416667  mrr: 0.416667  coverage_error: 2.500000
↳ ranking_loss: 0.750000  len: 3.000000
support: 2
>>> ranking_report(y_true, y_pred, bottom=True) # doctest: +NORMALIZE_WHITESPACE
      ndcg@k  precision@k  recall@k  f1@k  len@k  support@k  ndcg@k(B) \
1  1.000000  0.000000  0.0  0.0  1.0  2  1.000000
3  0.565465  0.333333  1.0  0.5  3.0  2  0.806574
5  0.565465  0.333333  1.0  0.5  3.0  2  0.806574
10 0.565465  0.333333  1.0  0.5  3.0  2  0.806574
<BLANKLINE>
      precision@k(B)  recall@k(B)  f1@k(B)  len@k(B)  support@k(B)
1  0.500000  0.25  0.333333  1.0  2
3  0.666667  1.00  0.800000  3.0  2
5  0.666667  1.00  0.800000  3.0  2
10 0.666667  1.00  0.800000  3.0  2
auc: 0.250000      map: 0.416667  mrr: 0.416667  coverage_error: 2.500000
↳ ranking_loss: 0.750000  len: 3.000000
support: 2  map(B): 0.708333  mrr(B): 0.750000
>>> ranking_report(y_true, y_pred, bottom=True, metrics=["auc"]) # doctest:
↳ +NORMALIZE_WHITESPACE
auc: 0.250000  len: 3.000000  support: 2
>>> y_true = [[0.9, 0.7, 0.1], [0, 0.5, 1]]
>>> y_pred = [[0.75, 0.5, 1], [1, 0.2, 0.1]]
>>> ranking_report(y_true, y_pred, continuous=True) # doctest: +NORMALIZE_
↳ WHITESPACE
      ndcg@k  len@k  support@k
3  0.675647  3.0  2
5  0.675647  3.0  2
10 0.675647  3.0  2
mrr: 0.750000  len: 3.000000  support: 2
>>> y_true = [[1, 0], [0, 0, 1]]
>>> y_pred = [[0.75, 0.5], [1, 0.2, 0.1]]
>>> ranking_report(y_true, y_pred) # doctest: +NORMALIZE_WHITESPACE
      ndcg@k  precision@k  recall@k  f1@k  len@k  support@k
1  1.00  0.500000  0.5  0.500000  1.0  2
3  0.75  0.416667  1.0  0.583333  2.5  2
5  0.75  0.416667  1.0  0.583333  2.5  2
10 0.75  0.416667  1.0  0.583333  2.5  2
auc: 0.500000      map: 0.666667  mrr: 0.666667  coverage_error: 2.000000
↳ ranking_loss: 0.500000  len: 2.500000
support: 2
>>> ranking_report(y_true, y_pred, coerce="abandon") # doctest: +NORMALIZE_
↳ WHITESPACE
      ndcg@k  precision@k  recall@k  f1@k  len@k  support@k
1  1.0  0.500000  0.5  0.5  1.0  2

```

()

()

```

3      0.5      0.333333      1.0      0.5      3.0      1
auc: 0.500000      map: 0.666667      mrr: 0.666667      coverage_error: 2.000000
↳ ranking_loss: 0.500000      len: 2.500000
support: 2
>>> ranking_report(y_true, y_pred, coerce="padding") # doctest: +NORMALIZE_
↳ WHITESPACE
      ndcg@k      precision@k      recall@k      f1@k      len@k      support@k
1      1.00      0.500000      0.5      0.500000      1.0      2
3      0.75      0.416667      1.0      0.583333      2.5      2
5      0.75      0.416667      1.0      0.583333      2.5      2
10     0.75      0.416667      1.0      0.583333      2.5      2
auc: 0.500000      map: 0.666667      mrr: 0.666667      coverage_error: 2.000000
↳ ranking_loss: 0.500000      len: 2.500000
support: 2
>>> ranking_report(y_true, y_pred, bottom=True) # doctest: +NORMALIZE_WHITESPACE
      ndcg@k      precision@k      recall@k      f1@k      len@k      support@k      ndcg@k(B)  \
1      1.00      0.500000      0.5      0.500000      1.0      2      1.000000
3      0.75      0.416667      1.0      0.583333      2.5      2      0.846713
5      0.75      0.416667      1.0      0.583333      2.5      2      0.846713
10     0.75      0.416667      1.0      0.583333      2.5      2      0.846713
<BLANKLINE>
      precision@k(B)      recall@k(B)      f1@k(B)      len@k(B)      support@k(B)
1      0.500000      0.5      0.500000      1.0      2
3      0.583333      1.0      0.733333      2.5      2
5      0.583333      1.0      0.733333      2.5      2
10     0.583333      1.0      0.733333      2.5      2
auc: 0.500000      map: 0.666667      mrr: 0.666667      coverage_error: 2.000000
↳ ranking_loss: 0.500000      len: 2.500000
support: 2      map(B): 0.791667      mrr(B): 0.750000
>>> ranking_report(y_true, y_pred, bottom=True, coerce="abandon") # doctest:
↳ +NORMALIZE_WHITESPACE
      ndcg@k      precision@k      recall@k      f1@k      len@k      support@k      ndcg@k(B)  \
1      1.0      0.500000      0.5      0.5      1.0      2      1.000000
3      0.5      0.333333      1.0      0.5      3.0      1      0.693426
<BLANKLINE>
      precision@k(B)      recall@k(B)      f1@k(B)      len@k(B)      support@k(B)
1      0.500000      0.5      0.5      1.0      2
3      0.666667      1.0      0.8      3.0      1
auc: 0.500000      map: 0.666667      mrr: 0.666667      coverage_error: 2.000000
↳ ranking_loss: 0.500000
len: 2.500000      support: 2      map(B): 0.791667      mrr(B): 0.750000
>>> ranking_report(y_true, y_pred, bottom=True, coerce="padding") # doctest:
↳ +NORMALIZE_WHITESPACE
      ndcg@k      precision@k      recall@k      f1@k      len@k      support@k      ndcg@k(B)  \
1      1.00      0.500000      0.5      0.500000      1.0      2      1.000000
3      0.75      0.416667      1.0      0.583333      2.5      2      0.846713
5      0.75      0.416667      1.0      0.583333      2.5      2      0.846713
10     0.75      0.416667      1.0      0.583333      2.5      2      0.846713
<BLANKLINE>
      precision@k(B)      recall@k(B)      f1@k(B)      len@k(B)      support@k(B)
1      0.50      0.5      0.500000      1.0      2
3      0.50      1.0      0.650000      3.0      2
5      0.30      1.0      0.452381      5.0      2
10     0.15      1.0      0.257576      10.0      2
auc: 0.500000      map: 0.666667      mrr: 0.666667      coverage_error: 2.000000
↳ ranking_loss: 0.500000      len: 2.500000
support: 2      map(B): 0.791667      mrr(B): 0.750000

```

3.5.3 toolkit

Monitor

epochbatch progress

monitor group monitor

API reference

General Toolkit

analyser

```
longling.ML.toolkit.analyser.get_max(src: ((<class 'str'>, <class 'pathlib.PurePath'>),  
                                         <class 'list'>), *keys, with_keys: (<class 'str'>, None)  
                                         = None, with_all=False, merge=True)
```

```
>>> src = [  
... {"Epoch": 0, "macro avg": {"f1": 0.7}, "loss": 0.04, "accuracy": 0.7},  
... {"Epoch": 1, "macro avg": {"f1": 0.88}, "loss": 0.03, "accuracy": 0.8},  
... {"Epoch": 1, "macro avg": {"f1": 0.7}, "loss": 0.02, "accuracy": 0.66}  
... ]  
>>> result, _ = get_max(src, "accuracy", merge=False)  
>>> result  
{'accuracy': 0.8}  
>>> _, result_appendix = get_max(src, "accuracy", with_all=True, merge=False)  
>>> result_appendix  
{'accuracy': {'Epoch': 1, 'macro avg': {'f1': 0.88}, 'loss': 0.03, 'accuracy': 0.  
↪8}}  
>>> result, result_appendix = get_max(src, "accuracy", "macro avg:f1", with_keys=  
↪"Epoch", merge=False)  
>>> result  
{'accuracy': 0.8, 'macro avg:f1': 0.88}  
>>> result_appendix  
{'accuracy': {'Epoch': 1}, 'macro avg:f1': {'Epoch': 1}}  
>>> get_max(src, "accuracy", "macro avg:f1", with_keys="Epoch")  
{'accuracy': {'Epoch': 1, 'accuracy': 0.8}, 'macro avg:f1': {'Epoch': 1, 'macro_  
↪avg:f1': 0.88}}
```

```
longling.ML.toolkit.analyser.get_min(src: ((<class 'str'>, <class 'pathlib.PurePath'>),  
                                         <class 'list'>), *keys, with_keys: (<class 'str'>, None)  
                                         = None, with_all=False, merge=True)
```

```
>>> src = [  
... {"Epoch": 0, "macro avg": {"f1": 0.7}, "loss": 0.04, "accuracy": 0.7},  
... {"Epoch": 1, "macro avg": {"f1": 0.88}, "loss": 0.03, "accuracy": 0.8},  
... {"Epoch": 1, "macro avg": {"f1": 0.7}, "loss": 0.02, "accuracy": 0.66}  
... ]  
>>> get_min(src, "loss")  
{'loss': 0.02}
```


`longling.ML.toolkit.analyser.key_parser(key)`

```
>>> key_parser("macro avg:f1")
['macro avg', 'f1']
>>> key_parser("accuracy")
'accuracy'
>>> key_parser("iteration:accuracy")
['iteration', 'accuracy']
```

dataset

class `longling.ML.toolkit.dataset.ID2Feature` (*feature_df: pandas.core.frame.DataFrame, id_field=None, set_index=False*)

```
>>> import pandas as pd
>>> df = pd.DataFrame({"id": [0, 1, 2, 3, 4], "numeric": [1, 2, 3, 4, 5], "text": ["a", "b", "c", "d", "e"]})
>>> i2f = ID2Feature(df, id_field="id", set_index=True)
>>> i2f[2]
numeric      3
text         c
Name: 2, dtype: object
>>> i2f[[2, 3]]["numeric"]
id
2      3
3      4
Name: numeric, dtype: int64
>>> i2f(2)
[3, 'c']
>>> i2f([2, 3])
[[3, 'c'], [4, 'd']]
```

class `longling.ML.toolkit.dataset.ItemSpecificSampler` (*triplet_df: pandas.core.frame.DataFrame, query_field='item_id', pos_field='pos', neg_field='neg', set_index=False, item_id_range=None, user_id_range=None, random_state=10*)

```
>>> import pandas as pd
>>> user_num = 3
>>> item_num = 4
```

()

```

>>> rating_matrix = pd.DataFrame({
...     "user_id": [0, 1, 1, 1, 2],
...     "item_id": [1, 3, 0, 2, 1]
... })
>>> triplet_df = ItemSpecificSampler.rating2triplet(rating_matrix)
>>> triplet_df      # doctest: +NORMALIZE_WHITESPACE
      pos neg
item_id
0      [1]  []
1      [0, 2] []
2      [1]  []
3      [1]  []
>>> triplet_df.index
Int64Index([0, 1, 2, 3], dtype='int64', name='item_id')
>>> sampler = ItemSpecificSampler(triplet_df)
>>> sampler(1)
(0, [0])
>>> sampler = ItemSpecificSampler(triplet_df, user_id_range=user_num)
>>> sampler(0, implicit=True)
(1, [2])
>>> sampler(0, 5, implicit=True)
(2, [2, 0, 0, 0, 0])
>>> sampler(0, 5, implicit=True, pad_value=-1)
(2, [0, 2, -1, -1, -1])
>>> sampler([0, 1, 2], 5, implicit=True, pad_value=-1)
[(2, [0, 2, -1, -1, -1]), (1, [1, -1, -1, -1, -1]), (2, [0, 2, -1, -1, -1])]
>>> rating_matrix = pd.DataFrame({
...     "user_id": [0, 1, 1, 1, 2],
...     "item_id": [1, 3, 0, 2, 1],
...     "score": [1, 0, 1, 1, 0]
... })
>>> triplet_df = ItemSpecificSampler.rating2triplet(rating_matrix,
↳value_field="score")
>>> triplet_df      # doctest: +NORMALIZE_WHITESPACE
      pos neg
item_id
0      [1]  []
1      [0]  [2]
2      [1]  []
3      []  [1]
>>> sampler = UserSpecificPairSampler(triplet_df)
>>> sampler([0, 1, 2], 5, pad_value=-1)
[(0, [-1, -1, -1, -1, -1]), (1, [2, -1, -1, -1, -1]), (0, [-1, -1, -1, -1, -1])]
>>> sampler([0, 1, 2], 5, neg=False, pad_value=-1)
[(1, [1, -1, -1, -1, -1]), (1, [0, -1, -1, -1, -1]), (1, [1, -1, -1, -1, -1])]
>>> sampler(rating_matrix["item_id"], 2, neg=rating_matrix["score"],
...     excluded_key=rating_matrix["user_id"], pad_value=-1)
[(1, [2, -1]), (0, [-1, -1]), (0, [-1, -1]), (0, [-1, -1]), (1, [0, -1])]
>>> sampler(rating_matrix["item_id"], 2, neg=rating_matrix["score"],
...     excluded_key=rating_matrix["user_id"], pad_value=-1, return_column=True)
((1, 0, 0, 0, 1), ([2, -1], [-1, -1], [-1, -1], [-1, -1], [0, -1]))
>>> sampler(rating_matrix["item_id"], 2, neg=rating_matrix["score"],
...     excluded_key=rating_matrix["user_id"], pad_value=-1, return_column=True,
↳split_sample_to_column=True)
((1, 0, 0, 0, 1), [(2, -1, -1, -1, 0), (-1, -1, -1, -1, -1)])

```

```

class longling.ML.toolkit.dataset.TripletPairSampler(triplet_df: pandas.core.frame.DataFrame,
query_field, pos_field='pos',
neg_field='neg',
set_index=False, query_range:
(<class 'int'>, <class 'tuple'>, <class 'list'>) = None,
key_range: (<class 'int'>,
<class 'tuple'>, <class 'list'>)
= None, random_state=10)

```

```

>>> # implicit feedback
>>> import pandas as pd
>>> triplet_df = pd.DataFrame({
...     "query": [0, 1, 2],
...     "pos": [[1], [3, 0, 2], [1]],
...     "neg": [[], [], []]
... })
>>> sampler = TripletPairSampler(triplet_df, "query", set_index=True)
>>> rating_matrix = pd.DataFrame({
...     "query": [0, 1, 1, 1, 2],
...     "key": [1, 3, 0, 2, 1]
... })
>>> triplet_df = TripletPairSampler.rating2triplet(rating_matrix, query_field=
↪"query", key_field="key")
>>> triplet_df # doctest: +NORMALIZE_WHITESPACE
      pos neg
query
0      [1] []
1    [3, 0, 2] []
2      [1] []
>>> sampler = TripletPairSampler(triplet_df, "query")
>>> sampler(0)
(0, [0])
>>> sampler(0, 3)
(0, [0, 0, 0])
>>> sampler(0, 3, padding=False)
(0, [])
>>> sampler = TripletPairSampler(triplet_df, "query", query_range=3, key_range=4)
>>> sampler(0)
(0, [0])
>>> sampler(0, 3)
(0, [0, 0, 0])
>>> sampler(0, 3, padding=False)
(0, [])
>>> sampler(0, 5, padding=False, implicit=True)
(3, [2, 3, 0])
>>> sampler(0, 5, padding=False, implicit=True, excluded_key=[3])
(2, [0, 2])
>>> sampler(0, 5, padding=True, implicit=True, excluded_key=[3])
(2, [2, 0, 0, 0, 0])
>>> sampler(0, 5, implicit=True, pad_value=-1)
(3, [2, 3, 0, -1, -1])
>>> sampler(0, 5, implicit=True, fast_implicit=True, pad_value=-1)

```

()

```

(3, [0, 2, 3, -1, -1])
>>> sampler(0, 5, implicit=True, fast_implicit=True, with_n_implicit=3, pad_
↪value=-1)
(3, [0, 2, 3, -1, -1, -1, -1, -1])
>>> sampler(0, 5, implicit=True, fast_implicit=True, with_n_implicit=3, pad_
↪value=-1, padding_implicit=True)
(3, [0, 2, 3, -1, -1, -1, -1, -1])
>>> rating_matrix = pd.DataFrame({
...     "query": [0, 1, 1, 1, 2],
...     "key": [1, 3, 0, 2, 1],
...     "score": [1, 0, 1, 1, 0]
... })
>>> triplet_df = TripletPairSampler.rating2triplet(
...     rating_matrix,
...     "query", "key",
...     value_field="score"
... )
>>> triplet_df      # doctest: +NORMALIZE_WHITESPACE
      pos  neg
query
0      [1]   []
1     [0, 2]  [3]
2      []   [1]
>>> sampler = TripletPairSampler(triplet_df, "query", query_range=3, key_range=4)
>>> sampler([0, 1, 2], 5, implicit=True, pad_value=-1)
[(3, [2, 3, 0, -1, -1]), (1, [1, -1, -1, -1, -1]), (3, [3, 0, 2, -1, -1])]
>>> sampler([0, 1, 2], 5, pad_value=-1)
[(0, [-1, -1, -1, -1, -1]), (1, [3, -1, -1, -1, -1]), (1, [1, -1, -1, -1, -1])]
>>> sampler([0, 1, 2], 5, neg=False, pad_value=-1)
[(1, [1, -1, -1, -1, -1]), (2, [0, 2, -1, -1, -1]), (0, [-1, -1, -1, -1, -1])]
>>> sampler(rating_matrix["query"], 2, neg=rating_matrix["score"],
...     excluded_key=rating_matrix["key"], pad_value=-1)
[(0, [-1, -1]), (2, [2, 0]), (1, [3, -1]), (1, [3, -1]), (0, [-1, -1])]
>>> sampler(rating_matrix["query"], 2, neg=rating_matrix["score"],
...     excluded_key=rating_matrix["key"], pad_value=-1, return_column=True)
((0, 2, 1, 1, 0), ([-1, -1], [0, 2], [3, -1], [3, -1], [-1, -1]))
>>> sampler(rating_matrix["query"], 2, neg=rating_matrix["score"],
...     excluded_key=rating_matrix["key"], pad_value=-1, return_column=True,
↪split_sample_to_column=True)
((0, 2, 1, 1, 0), ([-1, 0, 3, 3, -1], (-1, 2, -1, -1, -1)))
>>> rating_matrix = pd.DataFrame({
...     "query": [0, 1, 1, 1, 2],
...     "key": [1, 3, 0, 2, 1],
...     "score": [0.8, 0.4, 0.7, 0.5, 0.1]
... })
>>> TripletPairSampler.rating2triplet(
...     rating_matrix,
...     "query", "key",
...     value_field="score",
...     value_threshold=0.5
... )      # doctest: +NORMALIZE_WHITESPACE
      pos  neg
query
0      [1]   []
1     [0, 2]  [3]
2      []   [1]

```

```

class longling.ML.toolkit.dataset.UserSpecificPairSampler (triplet_df: pandas.core.frame.DataFrame,
                                                          query_field='user_id',
                                                          pos_field='pos',
                                                          neg_field='neg',
                                                          set_index=False,
                                                          user_id_range=None,
                                                          item_id_range=None,
                                                          random_state=10)

```

```

>>> import pandas as pd
>>> user_num = 3
>>> item_num = 4
>>> rating_matrix = pd.DataFrame({
...     "user_id": [0, 1, 1, 1, 2],
...     "item_id": [1, 3, 0, 2, 1]
... })
>>> triplet_df = UserSpecificPairSampler.rating2triplet(rating_matrix)
>>> triplet_df # doctest: +NORMALIZE_WHITESPACE
      pos neg
user_id
0      [1]  []
1     [3, 0, 2] []
2      [1]  []
>>> sampler = UserSpecificPairSampler(triplet_df)
>>> sampler(1)
(0, [0])
>>> sampler = UserSpecificPairSampler(triplet_df, item_id_range=item_num)
>>> sampler(0, implicit=True)
(1, [3])
>>> sampler(0, 5, implicit=True)
(3, [3, 2, 0, 0, 0])
>>> sampler(0, 5, implicit=True, pad_value=-1)
(3, [3, 2, 0, -1, -1])
>>> sampler([0, 1, 2], 5, implicit=True, pad_value=-1)
[(3, [2, 3, 0, -1, -1]), (1, [1, -1, -1, -1, -1]), (3, [2, 0, 3, -1, -1])]
>>> rating_matrix = pd.DataFrame({
...     "user_id": [0, 1, 1, 1, 2],
...     "item_id": [1, 3, 0, 2, 1],
...     "score": [1, 0, 1, 1, 0]
... })
>>> triplet_df = UserSpecificPairSampler.rating2triplet(rating_matrix=rating_
↪matrix, value_field="score")
>>> triplet_df # doctest: +NORMALIZE_WHITESPACE
      pos neg
user_id
0      [1]  []
1     [0, 2] [3]
2      []  [1]
>>> sampler = UserSpecificPairSampler(triplet_df)
>>> sampler([0, 1, 2], 5, pad_value=-1)
[(0, [-1, -1, -1, -1, -1]), (1, [3, -1, -1, -1, -1]), (1, [1, -1, -1, -1, -1])]
>>> sampler([0, 1, 2], 5, neg=False, pad_value=-1)
[(1, [1, -1, -1, -1, -1]), (2, [0, 2, -1, -1, -1]), (0, [-1, -1, -1, -1, -1])]

```

0

```

>>> sampler(rating_matrix["user_id"], 2, neg=rating_matrix["score"],
...         excluded_key=rating_matrix["item_id"], pad_value=-1)
[(0, [-1, -1]), (2, [2, 0]), (1, [3, -1]), (1, [3, -1]), (0, [-1, -1])]
>>> sampler(rating_matrix["user_id"], 2, neg=rating_matrix["score"],
...         excluded_key=rating_matrix["item_id"], pad_value=-1, return_column=True)
((0, 2, 1, 1, 0), ([-1, -1], [0, 2], [3, -1], [3, -1], [-1, -1]))
>>> sampler(rating_matrix["user_id"], 2, neg=rating_matrix["score"],
...         excluded_key=rating_matrix["item_id"], pad_value=-1, return_column=True,
...         split_sample_to_column=True)
((0, 2, 1, 1, 0), ([-1, 2, 3, 3, -1], (-1, 0, -1, -1, -1)))

```

longling.ML.toolkit.dataset.**train_test** (*files, train_size: (<class 'float'>, <class 'int'>) = 0.8, test_size: (<class 'float'>, <class 'int'>, None) = None, ratio=None, random_state=None, shuffle=True, target_names=None, suffix: list = None, prefix="", logger=<Logger dataset (INFO)>, **kwargs)

- **files** –
- **train_size** (float, int, or None, (default=0.8)) – Represent the proportion of the dataset to include in the train split.
- **test_size** (float, int, or None) – Represent the proportion of the dataset to include in the train split.
- **random_state** (int, RandomState instance or None, optional (default=None)) – If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.
- **shuffle** (boolean, optional (default=True)) – Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None
- **target_names** (list of PATH_TYPE) –
- **suffix** (list) –
- **kwargs** –

longling.ML.toolkit.dataset.**train_valid_test** (*files, train_size: (<class 'float'>, <class 'int'>) = 0.8, valid_size: (<class 'float'>, <class 'int'>) = 0.1, test_size: (<class 'float'>, <class 'int'>, None) = None, ratio=None, random_state=None, shuffle=True, target_names=None, suffix: list = None, logger=<Logger dataset (INFO)>, prefix="", **kwargs)

- **files** –
- **train_size** (float, int, or None, (default=0.8)) – Represent the proportion of the dataset to include in the train split.
- **valid_size** (float, int, or None, (default=0.1)) – Represent the proportion of the dataset to include in the valid split.

- **test_size** (*float, int, or None*) – Represent the proportion of the dataset to include in the test split.
- **random_state** (*int, RandomState instance or None, optional (default=None)*) – If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.
- **shuffle** (*boolean, optional (default=True)*) – Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None
- **target_names** –
- **suffix** (*list*) –
- **kwargs** –

formatter

```
class longling.ML.toolkit.formatter.EpisodeEvalFMT (logger=<RootLogger      root
(WARNING)>, dump_file:
(((<class 'str'>, <class
'pathlib.PurePath'>), (<class
'_io.TextIOWrapper'>, <class
'typing.TextIO'>, <class
'typing.BinaryIO'>, <class
'codecs.StreamReaderWriter'>,
<class 'fileinput.FileInput'>)),
None) = False, col: (<class 'int'>,
None) = None, **kwargs)
```

```
>>> import numpy as np
>>> from longling.ML.metrics import classification_report
>>> y_true = np.array([0, 0, 1, 1, 2, 1])
>>> y_pred = np.array([2, 1, 0, 1, 1, 0])
>>> y_score = np.array([
...     [0.15, 0.4, 0.45],
...     [0.1, 0.9, 0.0],
...     [0.33333, 0.33333, 0.33333],
...     [0.15, 0.4, 0.45],
...     [0.1, 0.9, 0.0],
...     [0.33333, 0.33333, 0.33333]
... ])
>>> print(EpisodeEvalFMT.format(
...     iteration=30,
...     eval_name_value=classification_report(y_true, y_pred, y_score)
... )) # doctest: +NORMALIZE_WHITESPACE
Episode [30]
      precision    recall  f1   support
0      0.000000  0.000000  0.000000         2
1      0.333333  0.333333  0.333333         3
2      0.000000  0.000000  0.000000         1
macro_avg  0.111111  0.111111  0.111111         6
accuracy: 0.166667  macro_auc: 0.194444
```

```
class longling.ML.toolkit.formatter.EpochEvalFMT (logger=<RootLogger root (WARN-
ING)>, dump_file: (((<class 'str'>,
<class 'pathlib.PurePath'>), (<class
'_io.TextIOWrapper'>, <class 'typ-
ing.TextIO'>, <class 'typing.BinaryIO'>,
<class 'codecs.StreamReaderWriter'>,
<class 'fileinput.FileInput'>)), None) =
False, col: (<class 'int'>, None) =
None, **kwargs)
```

```
>>> import numpy as np
>>> from longling.ML.metrics import classification_report
>>> y_true = np.array([0, 0, 1, 1, 2, 1])
>>> y_pred = np.array([2, 1, 0, 1, 1, 0])
>>> y_score = np.array([
...     [0.15, 0.4, 0.45],
...     [0.1, 0.9, 0.0],
...     [0.333333, 0.333333, 0.333333],
...     [0.15, 0.4, 0.45],
...     [0.1, 0.9, 0.0],
...     [0.333333, 0.333333, 0.333333]
... ])
>>> print(EpochEvalFMT.format(
...     iteration=30,
...     eval_name_value=classification_report(y_true, y_pred, y_score)
... )) # doctest: +NORMALIZE_WHITESPACE
Epoch [30]
      precision    recall  f1   support
0      0.000000   0.000000  0.000000         2
1      0.333333   0.333333  0.333333         3
2      0.000000   0.000000  0.000000         1
macro_avg  0.111111  0.111111  0.111111         6
accuracy: 0.166667  macro_auc: 0.194444
```

```
class longling.ML.toolkit.formatter.EvalFMT (logger=<RootLogger root (WARN-
ING)>, dump_file: (((<class 'str'>,
<class 'pathlib.PurePath'>), (<class
'_io.TextIOWrapper'>, <class 'typ-
ing.TextIO'>, <class 'typing.BinaryIO'>,
<class 'codecs.StreamReaderWriter'>,
<class 'fileinput.FileInput'>)), None) =
False, col: (<class 'int'>, None) = None,
**kwargs)
```

- **logger** – root logger
- **dump_file** – dump_file
- **col** (*int*) –
- **kwargs** –


```

>>> import numpy as np
>>> from longling.ML.metrics import classification_report
>>> y_true = np.array([0, 0, 1, 1, 2, 1])
>>> y_pred = np.array([2, 1, 0, 1, 1, 0])
>>> y_score = np.array([
...     [0.15, 0.4, 0.45],
...     [0.1, 0.9, 0.0],
...     [0.33333, 0.33333, 0.33333],
...     [0.15, 0.4, 0.45],
...     [0.1, 0.9, 0.0],
...     [0.33333, 0.33333, 0.33333]
... ])
>>> print(EvalFMT.format(
...     iteration=30,
...     eval_name_value=classification_report(y_true, y_pred, y_score)
... )) # doctest: +NORMALIZE_WHITESPACE
Iteration [30]
      precision    recall  f1   support
0      0.000000  0.000000  0.000000         2
1      0.333333  0.333333  0.333333         3
2      0.000000  0.000000  0.000000         1
macro_avg  0.111111  0.111111  0.111111         6
accuracy: 0.166667  macro_auc: 0.194444

```

`longling.ML.toolkit.formatter.result_format` (*data: dict, col=None*)

- **data** –
- **col** –

```

>>> print(result_format({"a": 1, "b": 2})) # doctest: +NORMALIZE_WHITESPACE
a: 1      b: 2
>>> print(result_format({"a": 1, "b": {"1": 0.1, "2": 0.3}, "c": {"1": 0.4, "2": 0.0}}))
      1      2
b  0.1  0.3
c  0.4  0.0
a: 1

```

monitor

class `longling.ML.toolkit.monitor.EMAValue` (*value_function_names: (<class 'list'>, <class 'dict'>), smoothing_constant=0.1, *args, **kwargs*)

Exponential moving average: smoothing to give progressively lower weights to older values.

$$\text{losses}[\text{name}] = (1 - c) \times \text{previous_value} + c \times \text{loss_value}$$

```

>>> ema = EMAValue(["12"])
>>> ema["12"]
nan
>>> ema("12", 100)
>>> ema("12", 1)
>>> ema["12"]
90.1
>>> list(ema.values())
[90.1]
>>> list(ema.keys())
['12']
>>> list(ema.items())
[('12', 90.1)]
>>> ema.reset()
>>> ema["12"]
nan
>>> ema = EMAValue(["11", "12"])
>>> ema["12"], ema["11"]
(nan, nan)
>>> ema.update({"11": 1, "12": 10})
>>> ema.update({"11": 10, "12": 100})
>>> ema["11"]
1.9
>>> ema["12"]
19.0
>>> ema = EMAValue(["11"], smoothing_constant=0.0)
>>> ema["11"]
nan
>>> ema.update({"11": 1})
>>> ema.update({"11": 10})
>>> ema["11"]
1.0
>>> ema = EMAValue(["11"], smoothing_constant=1.0)
>>> ema.update({"11": 1})
>>> ema.update({"11": 10})
>>> ema["11"]
10.0
>>> @as_tmt_value
... def mse_loss(a):
...     return a ** 2
>>> ema = EMAValue({"mse": mse_loss})
>>> ema["mse"]
nan
>>> mse_loss(1)
1
>>> ema["mse"]
1
>>> mse_loss(10)
100
>>> ema["mse"]
10.9
>>> ema = EMAValue({"mse": mse_loss})
>>> mse_loss(1)
1
>>> ema["mse"]
1
>>> ema.monitor_off("mse")

```

0

()

```

>>> ema.func
{}
>>> mse_loss(10)
100
>>> "mse" not in ema
True
>>> ema.monitor_on("mse", mse_loss)
>>> mse_loss(10)
100
>>> ema["mse"]
100

```

get_update_value (name: str, value: (<class 'float'>, <class 'int'>))

- **name** (str) – The name of the value to be updated
- **value** (int or float) – New value to include in EMA.

class longling.ML.toolkit.monitor.**MovingLoss** (value_function_names: (<class 'list'>, <class 'dict'>), smoothing_constant=0.1, *args, **kwargs)

```

>>> lm = MovingLoss(["l2"])
>>> lm.losses
{'l2': nan}
>>> lm("l2", 100)
>>> lm("l2", 1)
>>> lm["l2"]
90.1

```

longling.ML.toolkit.monitor.**as_tmt_loss** (loss_obj, loss2value=<function <lambda>>)

- **loss_obj** –
- **loss2value** –

```

>>> @as_tmt_loss
... def mse(v):
...     return v ** 2
>>> mse(2)
4

```

longling.ML.toolkit.monitor.**as_tmt_value** (value_obj, transform=<function <lambda>>)

- **value_obj** –
- **transform** –

```

>>> def loss_f(a):
...     return a
>>> loss_f(10)
10
>>> tmt_loss_f = as_tmt_value(loss_f)
>>> tmt_loss_f(10)
10
>>> @as_tmt_value
... def loss_f2(a):
...     return a
>>> loss_f2(10)
10

```

hyper_search

longling.ML.toolkit.hyper_search.**prepare_hyper_search**(*cfg_kwargs*: dict, *re-*
porthook=None, *fi-*
nal_reporthook=None,
primary_key=None,
max_key=True, *re-*
porter_cls=None, *with_keys*:
(*<class 'list'>*, *<class 'str'>*,
None) = None, *final_keys*:
(*<class 'list'>*, *<class 'str'>*,
None) = None, *dump*=False,
disable=False)

Updated in v1.3.18

nni package nni nni

```

cfg_kwargs, reporthook, final_reporthook, tag = prepare_hyper_search(
    cfg_kwargs, Configuration, reporthook, final_reporthook, primary_key="macro_
    ↪avg:f1"
)

_cfg = Configuration(**cfg_kwargs)
model = Model(_cfg)
...

for epoch in range(_cfg.begin_epoch, _cfg.end_epoch):
    for batch_data in dataset:
        train_model(batch_data)

    data = evaluate_model()
    reporthook(data)

final_reporthook()

```

- **cfg_kwargs** (*dict*) – cfg
- **reporthook** –
- **final_reporthook** –

- **primary_key** - , nni.report_intermediate_result nni.
report_final_result metric default
- **max_key** (*bool*) -
- **reporter_cls** -
- **with_keys** (*list or str*) - metric final report primary_key
- **final_keys** (*list or str*) - with_keys report result primary_key
- **dump** (*bool*) - True workspace workspace/nni.get_experiment_id() /
nni.get_trial_id() nni
- **disable** -
- **cfg_kwargs** (*dict*) - nni
- **reporhook** (*function*) - iteration nni.report_intermediate_result
- **final_reporhook** - iteration nni.report_final_result
- **dump** (*bool*) -

```
class CFG(Configuration):
    hyper_params = {"hidden_num": 100}
    learning_rate = 0.001
    workspace = ""

cfg_kwargs, reporhook, final_reporhook, dump = prepare_hyper_search(
    {"learning_rate": 0.1}, CFG, primary_key="macro_avg:f1", with_keys="accuracy"
)
# cfg_kwargs: {'learning_rate': 0.1}
```

when nni start (e.g., using `nni create --config _config.yml`), suppose in `_config.yml`:
and in `_search_space.json`

```
{
  "hidden_num": {"_type": "choice", "_value": [500, 600, 700, 835, 900]},
}
```

one of the return `cfg_kwargs` is `{'hyper_params': {'hidden_num': 50}, 'learning_rate': 0.1}`

3.5.4 MxnetHelper

helper

MxnetSymbolNDArraygetF

[Full documentation](#)

gallery

- layer: attentionhighway
- loss: pairwise-loss
- network: TextCNN

[Full documentation](#)

glue

[glue/ModelName](#)

[Full documentation](#)

toolkit

mxnet

- (cpu | gpu): ctx
- : optimizer_cfg
- : viz

[Full documentation](#)

API reference

Helper Functions

General Toolkit

ctx

embedding

optimizer_cfg

select_exp

Here are some frequently used regex expression for select in collect_params()

viz

exception `longling.ML.MxnetHelper.toolkit.viz.VizError`

Glue: Gluon Example

Introduction

Glue (Gluon Example) aims to generate a neural network model template of Mxnet-Gluon which can be quickly developed into a mature model. The source code is [here](#)

Installation

It is automatically installed when you installing longling package. The tutorial of installing can be found [here](#).

Tutorial

With glue, it is possible to quickly construct a model. A demo case can be referred in [‘here <>’_](#). And the model can be divided into several different functionalities:

- ETL(extract-transform-load): generate the data stream for model;
- Symbol()

Also, we call those variables like working directory, path to data, hyper parameters

Generate template files

Run the following commands to use glue:

```
# Create a full project including docs and requirements
glue --model_name ModelName
# Or, only create a network model template
glue --model_name ModelName --skip_top
```

The template files will be generate in current directory. To change the position of files, use `--directory` option to specify the location:

```
glue --model_name ModelName --directory LOCATION
```

For more help, run `glue --help`

Guidance to modify the template

Overview

Usually, the project template will consist of doc files and model files. Assume the project name by default is ModelName, then the directory of model files will have the same name, the directory tree is like:

```
ModelName (Project)
----docs
----ModelName (Model)
```

And in ModelName(Model), there are one template file named ModelName.py and a directory containing four sub-template files.

The directory tree is like:

```

ModelName/
├── __init__.py
├── ModelName.py
├── Module/
│   ├── __init__.py
│   ├── configuration.py
│   ├── etl.py
│   ├── module.py
│   ├── run.py
│   └── sym/
│       ├── __init__.py
│       ├── fit_eval.py
│       ├── net.py
│       └── viz.py

```

- The **'configuration.py <>'** defines the all parameters should be configured, like where to store the model parameters and configuration parameters, the hyper-parameters of the neural network.
- The **'etl.py <>'** defines the process of extract-transform-load, which is the definition of data processing.
- The **'module.py <>'** serves as a high-level wrapper for sym.py, which provides the well-written interfaces, like model persistence, batch loop, epoch loop and data pre-process on distributed computation.
- The **'sym.py <>'** is the minimal model can be directly used to train, evaluate, also supports visualization. But some higher-level operations are not supported for simplification and modulation, which are defined in module.py.

Data Stream

- extract: extract the data from data src .. code-block:: python


```

def extract(data_src): # load data from file, the data format is looked like: # feature, label features
    = [] labels = [] with open(data_src) as f:
        for line in f: feature, label = line.split() features.append(feature) labels.append(label)
    return features, labels

```
- transform: Convert the extracted into batch data. The pre-process like bucketing can be defined here.

```

from mxnet import gluon
def transform(raw_data, params):
    #
    # raw_data --> batch_data

    batch_size = params.batch_size

    return gluon.data.DataLoader(gluon.data.ArrayDataset(raw_data), batch_size)

```

- etl: combine the extract and transform together.

Model Construction

Usually, there are three level components need to be configured:

1. bottom: the network symbol and how to fit and eval it;

2. middle: the higher level to define the batch and epoch, also the initialization and persistence of model parameters.
3. top: the api of model

Bottom

configuration

Find the configuration.py and define the configuration variables that you need, for example:

- begin_epoch
- end_epoch
- batch_size

Also, the paths can be configured:

```
import longling.ML.MxnetHelper.glue.parser as parser
from longling.ML.MxnetHelper.glue.parser import var2exp
import pathlib
import datetime

#
class Configuration(parser.Configuration):
    model_name = str(pathlib.Path(__file__).parents[1].name)

    root = pathlib.Path(__file__).parents[2]
    dataset = ""
    timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
    workspace = ""

    root_data_dir = "$root/data/$dataset" if dataset else "$root/data"
    data_dir = "$root_data_dir/data"
    root_model_dir = "$root_data_dir/model/$model_name"
    model_dir = "$root_model_dir/$workspace" if workspace else root_model_dir
    cfg_path = "$model_dir/configuration.json"

    def __init__(self, params_json=None, **kwargs):
        # set dataset
        if kwargs.get("dataset"):
            kwargs["root_data_dir"] = "$root/data/$dataset"
        # set workspace
        if kwargs.get("workspace"):
            kwargs["model_dir"] = "$root_model_dir/$workspace"

        # rebuild relevant directory or file path according to the kwargs
        _dirs = [
            "workspace", "root_data_dir", "data_dir", "root_model_dir",
            "model_dir"
        ]
        for _dir in _dirs:
            exp = var2exp(
                kwargs.get(_dir, getattr(self, _dir)),
                env_wrap=lambda x: "self.%s" % x
            )
            setattr(self, _dir, eval(exp))
```

How the variable paths work can be referred in [‘here <>’_](#)

Refer to the [prototype](#) for illustration. Refer to [‘full documents about Configuration <>’_](#) for details.

build the network symbol and test it

The network symbol file is [‘sym.py <>’_](#)

The following variables and functions should be rewritten (marked as *****todo*****):

two ways can be used to check whether the network works well:

1. Visualization:

- functions name: net_viz *
-

1. Numerical:

- ********

Gallery

layer

loss

network

I

- `longling.Architecture.cli.main`, 33
- `longling.Architecture.install_file`, 33
- `longling.Architecture.utils`, 35
- `longling.lib.candylib`, 10
- `longling.lib.clock`, 11
- `longling.lib.concurrency`, 12
- `longling.lib.formatter`, 13
- `longling.lib.iterator`, 15
- `longling.lib.loading`, 16
- `longling.lib.parser`, 18
- `longling.lib.path`, 23
- `longling.lib.progress`, 24
- `longling.lib.regex`, 25
- `longling.lib.stream`, 26
- `longling.lib.structure`, 29
- `longling.lib.testing`, 30
- `longling.lib.time`, 30
- `longling.lib.utilog`, 31
- `longling.lib.yaml_helper`, 31
- `longling.ML.metrics.classification`, 37
- `longling.ML.metrics.ranking`, 41
- `longling.ML.metrics.regression`, 39
- `longling.ML.MxnetHelper.toolkit.ctx`, 58
- `longling.ML.MxnetHelper.toolkit.select_exp`, 58
- `longling.ML.MxnetHelper.toolkit.viz`, 58
- `longling.ML.toolkit.analyser`, 44
- `longling.ML.toolkit.dataset`, 45
- `longling.ML.toolkit.formatter`, 51
- `longling.ML.toolkit.hyper_search`, 56
- `longling.ML.toolkit.monitor`, 53
- `longling.spider.download_data`, 32
- `longling.spider.lib.get_html`, 32

S

`StreamError`, 28

V

`VizError`, 58