datawork

Release 0.0.7.dev48+ga5c279a.d20190918

Sep 18, 2019

Examples

CHAPTER 1

datawork.api.config

Basic Option and Config functionality.

class datawork.api.config.**Config**(*options*)
A 'Config' is a collection of 'Option's.

__init__(*options*)
Construct a Config with given list of 'Option's.

__setattr__(*name*, *val*)
Overload setattr to only allow setting of listed options.

__str__()
Convert to string by listing all options.

get_hash()
Return hash of the dictionary representation of this 'Config'.

parents()
Return empty list of parents.

to_dict()
Convert to dictionary to prepare for JSON conversion.

class datawork.api.config.**Configurable**
A Configurable class contains a 'Config' attribute called '.config'.

__init__()
Set a default .config using the class attribute '.OPTIONS'.

parents()
Return *.config* as the only parent.

class datawork.api.config.**Option**(*desc=None*, *name=None*, *required=False*, *default=None*)
An Option is a JSON serializable argument to a function.

__init__(*desc=None*, *name=None*, *required=False*, *default=None*)
Construct an 'Option' with name, default value, and description.

__str__()
Show value and name for this Option.

add_argument(*parser*)
Add an argument to an argparse 'ArgumentParser'.

get_value()
Get the set value or raise a ValueError.

set_value(*value*)
Implement a guarded setter for this Option type.

value
Get the set value or raise a ValueError.

CHAPTER 2

datawork.api.data

Module implementing abstract Data class.

class datawork.api.data.**Data**(*desc=None*, *name=None*)

Data placeholder class.

This class represents data that has either not yet been computed, or is furthermore not fully specified. Classes inheriting *Data* implement placeholders for specific data types, e.g. Pandas dataframes or numpy arrays.

Subclasses of *Data* are typically instantiated by invocations of *Tool*.

Thus *Data* and *Invocation* are connected and form the backbone of the computational graph, with *Tool* objects connected to *Invocation* as objects that can be configured.

Note that the *provider* attribute itself an *Invocation*, can be "partial", in which case the data object itself is callable. When called, arguments are passed to the provider which will create new invocations; potentially now non-partial ones.

__call__(**args*)

Enable calling for placeholder Data objects.

__init__(*desc=None*, *name=None*)

Construct a placeholder data object.

Parameters

- **desc** – a plain-text description of this data object
- **name** – a short-hand name for this data object

__repr__()

Represent data including provider and name.

static check_type(*value*)

Guard value to ensure it is of proper type.

classmethod constant(*val*, *name='constant'*)

Create a constant from appropriately typed variable.

data
Getter for data attribute.

get_data()
Getter for data attribute.

get_hash()
Return hash of provider if exists, or of data itself for constants.

missing_args()
Count number of missing arguments.

parents()
Return provider as only parent if it is set.

read(*filename*)
Read data from disk.

static serialize(*data*)
Convert data to string.

set_data(*value*, *cache=True*)
Setter for data attribute.

write(*filename*)
Write data to disk.

CHAPTER 3

datawork.api.graph

Node class and associated graph traversal functionality.

class datawork.api.graph.**Node**
Abstract node class.

ancestors()
Return ancestry tree of node.

This method works by recursively calling itself on all parents of each Node, which themselves are assumed to be Nodes.

Returns **anc** – list of (object, ancestor tree) pairs.

Return type list

parents()
Return list of parent 'None' objects upon which this object depends.

datawork.api.graph.**compute_dag**(*outputs*)
Compute a directed acyclic graph with the given outputs.

datawork.api.graph.**extract_config**(*g*)
Extract dictionary with all configuration options found in graph.

datawork.api.graph.**extract_inputs**(*g*)
Extract Data nodes that have no "Provides" in-neighbors.

datawork.api.graph.**extract_tools**(*g*)
Given a graph, pull out all configurable Tools.

datawork.api.graph.**fill_graph**(*g*, *o*)
Add given node to graph and all ancestors.

datawork.api.graph.**node_label**(*obj*)
Compute a node label for this object.

datawork.api.graph.**unique_objects**(*l*)
Given list of objects, ensure that they are all distinct python objects.

datawork.api.graph.**visualize**(*g*, *filename*, *outputs*)
Draw computation DAG using graphviz.

CHAPTER 4

datawork.api.invocation

Module implementing Tool and Invocation.

class datawork.api.invocation.**Invocation**(*tool*, *args*)

Called tool connecting input data to output data.

This class represents a *Tool* with fully or partially specified inputs, ready for computation and caching. It is responsible for providing cache identifiers for all its outputs.

__init__(*tool*, *args*)

Construct invocation object.

Parameters

- **tool** – the *Tool* object being invoked
- **args** – tuple of arguments, which are either *Data* objects or *None*, in which case a new placeholder type will be instantiated.

cache_identifier(*o*)

Return identifier for cache which combines name of output with hash.

cache_outputs()

Write outputs to cache.

get_hash()

Compute a hash of this invocation.

get_outputs()

Implement a getter for evaluating outputs on demand.

invoke(**args*)

Handle partial evaluation by invoking with more arguments.

The result of this method is another *Invocation*.

If the same arguments (meaning the same objects, identified by python id) are provided, the same invocation object is returned.

missing_args()
Count number of missing arguments.

o
Implement a getter for evaluating outputs on demand.

parents()
Invocations depend on tool and all arguments.

populate()
Compute the outputs by calling a tool's '.run()' method.

This method organizes all of the input *data.Data* and *config.Option* parameters and passes them to the static *tool.Tool.run()* method.

set_output(*name*, *value*, *cache=True*)
Set output, caching if requested.

CHAPTER 5

datawork.api.tool

Module implementing Tool and Invocation.

class datawork.api.tool.**Tool**
A class for composable tools.

This is the base class for configurable functions that transform *Data* objects.

__call__(**args*)
Let a tool act on some *Data* objects.

Calling a tool instance _invokes_ the tool, which results in an *Invocation* instance if the arguments are of class *Data*.

__init__()
Construct an object with a new configuration.

__repr__()
Return the name of the tool and its config.

get_hash()
Return a hash string for this tool and configuration.

static run(*cfg*)
Compute outputs.

This method is the meat of the *Tool* class.

version()
Return a version string for this tool.

Subclasses implement their own version methods.

datawork.api.tool.**tool**(*r*)
Quickly create a *Tool* class.

Given a function definition for a tool, create a class with the provided function as the class's *run()* method.

CHAPTER 6

datawork.instances.config

Common instances of Option including most JSON types.

class datawork.instances.config.**BoolOption**(*desc=None*, *name=None*, *required=False*, *default=None*)

A boolean option.

add_argument(*parser*)

Add an argument with an action to an argparse 'ArgumentParser'.

value_type

alias of builtins.bool

class datawork.instances.config.**EnumOption**(*desc*, *choices=None*, ***kwargs*)

An enum option represents a choice from a finite list.

__init__(*desc*, *choices=None*, ***kwargs*)

Construct option that records possible choices.

__str__()

Format string that shows choices.

set_value(*value*)

Restrict set values to choices.

value_type

alias of builtins.str

class datawork.instances.config.**FloatOption**(*desc=None*, *name=None*, *required=False*, *default=None*)

A single float option.

value_type

alias of builtins.float

class datawork.instances.config.**IntOption**(*desc=None*, *name=None*, *required=False*, *default=None*)

A single integer option.

value_type
alias of builtins.int

class datawork.instances.config.**RandomSeedOption**(*desc=None*, *name=None*, *required=False*, *default=None*)

An *IntOption* subclass specifically for random seeds.

This class makes it a bit easier to detect random seeds in large pipelines, which should make studying variability due to controllable (RNG) randomness straightforward.

class datawork.instances.config.**StringOption**(*desc=None*, *name=None*, *required=False*, *default=None*)

A string option.

value_type
alias of builtins.str

CHAPTER 7

datawork.instances.data

Instances of Data for common data payloads.

class datawork.instances.data.**FileData**(*desc=None*, *name=None*)
Base class for any disk-native data.

For example, SQLiteData will use this as a base class.

static check_type(*value*)
Check that value is a filename.

read(*filename*)
Read by setting the filename.

static serialize(*data*)
Simply return the filename.

write(*filename*)
Copy file to new location.

class datawork.instances.data.**JSONData**(*desc=None*, *name=None*)
A Data class for primitive JSON serializable types.

The so-called "primitive types" in JSON are:

- string
- numeric types
- object (in python this is a `dict`)
- array
- boolean
- null

In this class, hierarchies of the following types are supported:

- `bool`
- `dict`

- float
- int
- list
- None
- str

Note that although other types than these may be serializable in Python (by subclassing json.JSONEncoder), the primitive types can be serialized/deserialized unambiguously. For example, we do not support tuples, although the json module supports serializing them by casting them to lists.

static check_type(*value*)
Check that value is a hierarchy of primitive JSON types.

read(*filename*)
Read JSON text.

static serialize(*data*)
Convert to JSON text.

write(*filename*)
Write as JSON text.

class datawork.instances.data.**KerasModelData**(*desc=None*, *name=None*)
A Data class for Keras models.

static check_type(*value*)
Check that value is a keras.models.Model.

read(*filename*)
Read from HDF5.

write(*filename*)
Write to HDF5.

class datawork.instances.data.**PandasData**(**args*, ***kwargs*)
Data type for Pandas DataFrames and Series.

__init__(**args*, ***kwargs*)
Construct PandasData.

static check_type(*value*)
Check that value is a DataFrame or Series.

read(*filename*)
Read from msgpack.

static serialize(*data*)
Write to msgpack.

write(*filename*)
Write msgpack.

class datawork.instances.data.**TorchModelData**(*desc=None*, *name=None*)
A Data class for PyTorch models.

static check_type(*value*)
Check that value is a torch.nn.Module.

read(*filename*)
Load state dict and module class.

write (*filename*)
Write state dict and serialize module class.

CHAPTER 8

datawork.utils.cmdline

Unified command line interface for datawork pipelines.

datawork.utils.cmdline.**command_line**(**outputs*)

Given output Data objects, create a standard command line interface and execute it.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

Symbols

A

B

C

D

E

W