
Dazu

Release 0.1.0

Raphael Pinto

Feb 24, 2020

CONTENTS:

1	Getting Started	3
2	Prerequisites	5
3	Installing Development Environment	7
4	Code Style	9
5	Running the tests	11
6	Deployment	13
7	Built With	15
8	Contributing	17
9	Versioning	19
10	Authors	21
11	License	23
12	Acknowledgments	25
13	Welcome to Dazus documentation!	27
	Python Module Index	37
	Index	39

Dazu is a powerfull engine dialogue engine with two main parts: NLU and dialogue. The main objetive of this project is too use existing chatbots projects and uses it to develop your solution.

The name was inspired by [Dazu Ausubel](#) because the main objective of this project was to build a collaborative platform to maintain Bots for learning.

Inspired by Watson Assistant and Rasa.

GETTING STARTED

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes. See deployment for notes on how to deploy the project on a live system.

- Docker:

```
docker-compose up --build
```

- Editable mode:

```
pip install -e .  
cd examples/my-first-bot  
dazu train  
dazu run
```

- After that you should see this output:

```
dazu run  
* Serving Flask app "dazu.__main__" (lazy loading)  
* Environment: production  
  WARNING: Do not use the development server in a production environment.  
  Use a production WSGI server instead.  
* Debug mode: off  
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```


PREREQUISITES

- To build, first install all necessary dependencies:

All the dependencies can be find in `requirements.txt` and development in `requirements-dev.txt`.

INSTALLING DEVELOPMENT ENVIRONMENT

- A step by step installation guide:

1. Run these commands to install dazul in your python virtual env:

```
pip install -r requirements-dev.txt  
pip install -e .
```

1. Go to examples folder and start the project:

```
cd examples/my-first-bot  
dazu train  
dazu run
```

1. Have fun :rocket:

CODE STYLE

To ensure a standardized code style we use the formatter [black](#). To ensure our type annotations are correct we use the type checker [pytype](#). If your code is not formatted properly or doesn't type check, travis will fail to build.

4.1 Formatting

If you want to automatically format your code on every commit, you can use [pre-commit](#). Just install it via `pip install pre-commit` and execute `pre-commit install` in the root folder. This will add a hook to the repository, which reformats files on every commit.

If you want to set it up manually, install black via `pip install -r requirements-dev.txt`. To reformat files execute

```
make formatter
```

4.2 Type Checking

If you want to check types on the codebase, install pytype using `pip install -r requirements-dev.txt`. To check the types execute

```
make types
```


RUNNING THE TESTS

Still needed

DEPLOYMENT

Still needed

BUILT WITH

- [Python](#) - The main programing language used

CONTRIBUTING

Please read [CONTRIBUTING.md](#) for details on our code of conduct, and the process for submitting pull requests to us.

VERSIONING

We use [SemVer](#) for versioning. For the versions available, see the [tags on this repository](#).

AUTHORS

- **Raphael Pinto** - *Creator* - [ralphg6](#)

See also the list of [contributors](#) who participated in this project.

LICENSE

This project is licensed under the Apache-2.0 - see the [LICENSE](#) file for details

ACKNOWLEDGMENTS

- Arthur Temporim - [arthurTemporim](#)

WELCOME TO DAZU'S DOCUMENTATION!

13.1 dazu package

13.1.1 Subpackages

13.1.2 Submodules

13.1.3 dazu.config module

```
class dazu.config.DazuConfig (configuration_values: Optional[Dict[str, Any]] = None)
    Bases: object

    as_dict () → Dict[str, Any]

    property component_names

    for_component (index, defaults=None) → Dict[str, Any]

    get (key: str, default: Any = None) → Any

    items () → List[Any]

    override (config) → None

    set_component_attr (index, **kwargs) → None

    view () → str

exception dazu.config.InvalidConfigError (message: str)
    Bases: ValueError

    Raised if an invalid configuration is encountered.

dazu.config.component_config_from_pipeline (index: int, pipeline: List[Dict[str, Any]], de-
                                           faults: Optional[Dict[str, Any]] = None) →
                                           Dict[str, Any]

dazu.config.load (config: Union[str, Dict, None] = None, **kwargs: Any) → dazu.config.DazuConfig

dazu.config.override_defaults (defaults: Optional[Dict[str, Any]], custom: Optional[Dict[str,
                                           Any]]) → Dict[str, Any]
```

13.1.4 dazu.constants module

13.1.5 dazu.registry module

```
class dazu.registry.Registry
    Bases: object

    classmethod get (moduleName: str = None, moduleCls: Type[dazu.typing.module.Module] = None,
                    prefix: str = "", default=None) → Any

    classmethod getAdapter (config: dazu.config.DazuConfig, adapterName: str = None)

    modules = {'dialogue_simple': <class 'dazu.components.dialogue.simple.SimpleDialogue'>

    classmethod registry (module: Type[dazu.typing.module.Module], prefix: str = "")

    classmethod registryAdapter (adapter: Type[dazu.typing.module.Module])

dazu.registry.pipeline_template (s: str) → Optional[List[Dict[str, Any]]]
```

13.1.6 dazu.util module

```
dazu.util.json_to_string (obj: Any, **kwargs: Any) → str
```

13.1.7 dazu.version module

13.1.8 Module contents

13.2 dazu.adapters package

13.2.1 Subpackages

13.2.2 Submodules

13.2.3 dazu.adapters.adapter module

```
class dazu.adapters.adapter.Adapter
    Bases: dazu.typing.module.Module

    Adapter Class documentation

    abstract classmethod input (payload: Dict) → dazu.typing.message.Message

    abstract classmethod output (message: dazu.typing.message.Message) → Dict

    classmethod validate_data (payload: Dict) → bool
        validate_data method documentation

class dazu.adapters.adapter.MessageAdapter
    Bases: dazu.adapters.adapter.Adapter

    classmethod input (payload: Dict) → dazu.typing.message.Message

    classmethod name ()
        The name property is a function of the class - its __name__.

    classmethod output (message: dazu.typing.message.Message) → Dict
```

```
classmethod validate_data (payload: Dict) → bool
    validate_data method documentation
```

13.2.4 Module contents

13.3 dazu.adapters.adapters package

13.3.1 Submodules

13.3.2 dazu.adapters.adapters.google module

```
class dazu.adapters.adapters.google.GoogleAdapter
    Bases: dazu.adapters.adapter.Adapter

    classmethod input (payload: Dict) → dzu.typing.message.Message

    classmethod name ()
        The name property is a function of the class - its __name__.

    classmethod output (message: dzu.typing.message.Message) → Dict

    classmethod validate_data (payload: Dict) → bool
        validate_data method documentation
```

13.3.3 Module contents

13.4 dzu.components package

13.4.1 Subpackages

13.4.2 Submodules

13.4.3 dzu.components.component module

```
class dzu.components.component.Component (component_config: Optional[Dict[str, Any]] =
                                          None)
    Bases: dzu.typing.module.Module
```

A component is a message processing unit in a pipeline. Components are collected sequentially in a pipeline. Each component is called one after another. This holds for initialization, training, persisting and loading the components. If a component comes first in a pipeline, its methods will be called first. E.g. to process an incoming message, the `process` method of each component will be called. During the processing (as well as the training, persisting and initialization) components can pass information to other components. The information is passed to other components by providing attributes to the so called pipeline context. The pipeline context contains all the information of the previous components a component can use to do its own processing. For example, a featurizer component can provide features that are used by another component down the pipeline to do intent classification.

```
classmethod cache_key (component_meta: Dict[str, Any], model_metadata:
                        dzu.typing.model.Metadata) → Optional[str]
```

This key is used to cache components. If a component is unique to a model it should return `None`. Otherwise, an instantiation of the component will be reused for all models where the metadata creates the same key.

classmethod can_handle_language (*language: Hashable*) → bool

Check if component supports a specific language. This method can be overwritten when needed. (e.g. dynamically determine which language is supported.)

classmethod create (*component_config: Dict[str, Any], config: dazu.config.DazuConfig*) →
dazu.components.component.Component

Creates this component (e.g. before a training is started). Method can access all configuration parameters.

defaults = {}

language_list = None

classmethod load (*component_config: Dict[str, Any], model_dir: Optional[str]*
= None, model_metadata: Optional[Metadata] = None,
*cached_component: Optional[Component] = None, **kwargs: Any*) →
dazu.components.component.Component

Load this component from file. After a component has been trained, it will be persisted by calling *persist*. When the pipeline gets loaded again, this component needs to be able to restore itself. Components can rely on any context attributes that are created by `components.Component.create()` calls to components previous to this one.

partially_process (*message: dazu.typing.message.Message*) → *dazu.typing.message.Message*

Allows the component to process messages during training (e.g. external training data). The passed message will be processed by all components previous to this one in the pipeline.

persist (*file_name: str, model_dir: str*) → *Optional[Dict[str, Any]]*

Persist this component to disk for future loading.

prepare_partial_processing (*pipeline: List[Component], context: Dict[str, Any]*) → None

Sets the pipeline and context used for partial processing. The pipeline should be a list of components that are previous to this one in the pipeline and have already finished their training (and can therefore be safely used to process messages).

process (*message: dazu.typing.message.Message, **kwargs: Any*) → None

Process an incoming message. This is the components chance to process an incoming message. The component can rely on any context attribute to be present, that gets created by a call to `rasa.nlu.components.Component.create()` of ANY component and on any context attributes created by a call to `rasa.nlu.components.Component.process()` of components previous to this one.

provide_context () → *Optional[Dict[str, Any]]*

Initialize this component for a new pipeline This function will be called before the training is started and before the first message is processed using the interpreter. The component gets the opportunity to add information to the context that is passed through the pipeline during training and message parsing. Most components do not need to implement this method. It's mostly used to initialize framework environments like MITIE and spacy (e.g. loading word vectors for the pipeline).

provides = []

classmethod required_packages () → *List[str]*

Specify which python packages need to be installed to use this component, e.g. ["spacy"]. More specifically, these should be importable python package names e.g. *sklearn* and not package names in the dependencies sense e.g. *scikit-learn* This list of requirements allows us to fail early during training if a required package is not installed.

requires = []

train (*training_data: dazu.typing.training_data.TrainingData, cfg: dazu.config.DazuConfig, **kwargs:*
Any) → None

Train this component. This is the components chance to train itself provided with the training data. The component can rely on any context attribute to be present, that gets created by a call to `rasa.nlu`.

`components.Component.create()` of ANY component and on any context attributes created by a call to `rasa.nlu.components.Component.train()` of components previous to this one.

exception `dazu.components.component.UnsupportedLanguageError` (*component: str, language: str*)

Bases: `Exception`

Raised when a component is created but the language is not supported.

Parameters

- **component** – component name
- **language** – language that component doesn't support

13.4.4 `dazu.components.engine` module

class `dazu.components.engine.Engine` (*config: `dazu.config.DazuConfig`*)

Bases: `object`

components: `List[dazu.components.component.Component]` = []

pipeline: `List[Type[dazu.components.component.Component]]` = []

respond (*message, context={}*)

train ()

`dazu.components.engine.build_model_filename` (*idx, componentCls: Type[dazu.components.component.Component]*)

13.4.5 Module contents

13.5 `dazu.components.dialogue` package

13.5.1 Submodules

13.5.2 `dazu.components.dialogue.simple` module

class `dazu.components.dialogue.simple.SimpleDialogue` (*component_config: Optional[Dict[str, Any]] = None, dialog_nodes: Dict = None*)

Bases: `dazu.components.component.Component`

classmethod **load** (*meta: Dict[str, Any], model_dir: Optional[str] = None, model_metadata: Optional[Metadata] = None, cached_component: Optional[Component] = None, **kwargs: Any*) → `dazu.components.component.Component`

Load this component from file. After a component has been trained, it will be persisted by calling `persist`. When the pipeline gets loaded again, this component needs to be able to restore itself. Components can rely on any context attributes that are created by `components.Component.create()` calls to components previous to this one.

classmethod **name** ()

The name property is a function of the class - its `__name__`.

persist (*file_name: str, model_dir: str*) → `Optional[Dict[str, Any]]`

Persist this component to disk for future loading.

process (*message: dazu.typing.message.Message, **kwargs: Any*) → None

Process an incoming message. This is the components chance to process an incoming message. The component can rely on any context attribute to be present, that gets created by a call to `rasa.nlu.components.Component.create()` of ANY component and on any context attributes created by a call to `rasa.nlu.components.Component.process()` of components previous to this one.

train (*training_data: dazu.typing.training_data.TrainingData, cfg: dazu.config.DazuConfig, **kwargs: Any*) → None

Train this component. This is the components chance to train itself provided with the training data. The component can rely on any context attribute to be present, that gets created by a call to `rasa.nlu.components.Component.create()` of ANY component and on any context attributes created by a call to `rasa.nlu.components.Component.train()` of components previous to this one.

`dazu.components.dialogue.simple.evalCondition(condition, context, intent, entities)`

`dazu.components.dialogue.simple.get_dialog_anythinelse(dialog_nodes)`

`dazu.components.dialogue.simple.get_dialog_welcome(dialog_nodes)`

13.5.3 Module contents

13.6 dazu.components.nlu package

13.6.1 Submodules

13.6.2 dazu.components.nlu.simple module

class `dazu.components.nlu.simple.SimpleNLU` (*component_config: Optional[Dict[str, Any]] = None, intent_model: Dict = None*)

Bases: `dazu.components.component.Component`

classmethod `load` (*meta: Dict[str, Any], model_dir: Optional[str] = None, model_metadata: Optional[Metadata] = None, cached_component: Optional[Component] = None, **kwargs: Any*) → `dazu.components.component.Component`

Load this component from file. After a component has been trained, it will be persisted by calling `persist`. When the pipeline gets loaded again, this component needs to be able to restore itself. Components can rely on any context attributes that are created by `components.Component.create()` calls to components previous to this one.

classmethod `name` ()

The name property is a function of the class - its `__name__`.

persist (*file_name: str, model_dir: str*) → `Optional[Dict[str, Any]]`

Persist this component to disk for future loading.

process (*message: dazu.typing.message.Message, **kwargs: Any*) → None

Process an incoming message. This is the components chance to process an incoming message. The component can rely on any context attribute to be present, that gets created by a call to `rasa.nlu.components.Component.create()` of ANY component and on any context attributes created by a call to `rasa.nlu.components.Component.process()` of components previous to this one.

train (*training_data: dazu.typing.training_data.TrainingData, cfg: dazu.config.DazuConfig, **kwargs: Any*) → None

Train this component. This is the components chance to train itself provided with the training data. The component can rely on any context attribute to be present, that gets created by a call to `rasa.nlu.components.Component.create()` of ANY component and on any context attributes created by a call to `rasa.nlu.components.Component.train()` of components previous to this one.

`dazu.components.nlu.simple.simmilarity(a, b)`

`dazu.components.nlu.simple.tokenize(stopwords: List[str], sentence: str)`

13.6.3 Module contents

13.7 dazu.training_data package

13.7.1 Subpackages

13.7.2 Submodules

13.7.3 dazu.training_data.reader module

```
class dazu.training_data.reader.Reader
    Bases: dazu.typing.module.Module

    abstract classmethod load(config: dazu.config.DazuConfig) →
                                dazu.typing.training_data.TrainingData

    classmethod validate_data(config: dazu.config.DazuConfig, data: Dict) → bool
```

13.7.4 Module contents

13.8 dazu.training_data.formats package

13.8.1 Submodules

13.8.2 dazu.training_data.formats.json module

```
class dazu.training_data.formats.json.JsonReader
    Bases: dazu.training_data.reader.Reader

    classmethod load(config: dazu.config.DazuConfig) → dazu.typing.training_data.TrainingData
```

13.8.3 Module contents

13.9 dazu.typing package

13.9.1 Submodules

13.9.2 dazu.typing.message module

```
class dazu.typing.message.Message(text: str, context=None, data={}, time=None)
    Bases: object

    classmethod build(text=None, intents=None, entities=None, context=None, payload=None) →
                                dazu.typing.message.Message

    get(prop, default=None) → Any
```

`set (prop, info) → None`

13.9.3 dazu.typing.model module

`class dazu.typing.model.Metadata (data)`
Bases: object

`class dazu.typing.model.Model (data)`
Bases: object

13.9.4 dazu.typing.module module

`class dazu.typing.module.Module`
Bases: object

Metaclass with *name* class property

`classmethod name ()`
The name property is a function of the class - its `__name__`.

13.9.5 dazu.typing.training_data module

`class dazu.typing.training_data.TrainingData (data)`
Bases: object

13.9.6 Module contents

13.10 dazu.utils package

13.10.1 Submodules

13.10.2 dazu.utils.io module

`dazu.utils.io.create_directory (directory_path: str) → None`
Creates a directory and its super paths. Succeeds even if the path already exists.

`dazu.utils.io.create_directory_for_file (file_path: str) → None`
Creates any missing parent directories of this file path.

`dazu.utils.io.create_path (file_path: str) → None`
Makes sure all directories in the 'file_path' exists.

`dazu.utils.io.create_temporary_file (data: Any, suffix: str = '', mode: str = 'w+') → str`
Creates a `tempfile.NamedTemporaryFile` object for data. mode defines `NamedTemporaryFile`'s mode parameter in `py3`.

`dazu.utils.io.dump_obj_as_json_to_file (filename: str, obj: Any) → None`
Dump an object as a json string to a file.

`dazu.utils.io.enable_async_loop_debugging (event_loop: asyncio.events.AbstractEventLoop,
slow_callback_duration: float = 0.1) → asyncio.events.AbstractEventLoop`

`dazu.utils.io.fix_yaml_loader()` → None
Ensure that any string read by yaml is represented as unicode.

`dazu.utils.io.is_subdirectory(path: str, potential_parent_directory: str)` → bool

`dazu.utils.io.list_directory(path: str)` → List[str]
Returns all files and folders excluding hidden files. If the path points to a file, returns the file. This is a recursive implementation returning files in any depth of the path.

`dazu.utils.io.list_files(path: str)` → List[str]
Returns all files excluding hidden files. If the path points to a file, returns the file.

`dazu.utils.io.list_subdirectories(path: str)` → List[str]
Returns all folders excluding hidden files. If the path points to a file, returns an empty list.

`dazu.utils.io.read_config_file(filename: str)` → Dict[str, Any]
Parses a yaml configuration file. Content needs to be a dictionary :param filename: The path to the file which should be read.

`dazu.utils.io.read_file(filename: str, encoding: str = 'utf-8')` → Any
Read text from a file.

`dazu.utils.io.read_json_file(filename: str)` → Any
Read json from a file.

`dazu.utils.io.read_yaml(content: str)` → Union[List[Any], Dict[str, Any]]
Parses yaml from a text. :param content: A text containing yaml content.

`dazu.utils.io.read_yaml_file(filename: str)` → Union[List[Any], Dict[str, Any]]
Parses a yaml file. :param filename: The path to the file which should be read.

`dazu.utils.io.replace_environment_variables()` → None
Enable yaml loader to process the environment variables in the yaml.

`dazu.utils.io.unarchive(byte_array: bytes, directory: str)` → str
Tries to unpack a byte array interpreting it as an archive. Tries to use tar first to unpack, if that fails, zip will be used.

`dazu.utils.io.write_text_file(content: str, file_path: Union[str, pathlib.Path], encoding: str = 'utf-8', append: bool = False)` → None
Writes text to a file. :param content: The content to write. :param file_path: The path to which the content should be written. :param encoding: The encoding which should be used. :param append: Whether to append to the file or to truncate the file.

`dazu.utils.io.write_yaml_file(data: Dict, filename: Union[str, pathlib.Path])` → None
Writes a yaml file. :param data: The data to write. :param filename: The path to the file which should be written.

`dazu.utils.io.zip_folder(folder: str)` → str
Create an archive from a folder.

13.10.3 Module contents

PYTHON MODULE INDEX

d

- dazu, [28](#)
- dazu.adapters, [29](#)
- dazu.adapters.adapter, [28](#)
- dazu.adapters.adapters, [29](#)
- dazu.adapters.adapters.google, [29](#)
- dazu.components, [31](#)
- dazu.components.component, [29](#)
- dazu.components.dialogue, [32](#)
- dazu.components.dialogue.simple, [31](#)
- dazu.components.engine, [31](#)
- dazu.components.nlu, [33](#)
- dazu.components.nlu.simple, [32](#)
- dazu.config, [27](#)
- dazu.constants, [28](#)
- dazu.registry, [28](#)
- dazu.training_data, [33](#)
- dazu.training_data.formats, [33](#)
- dazu.training_data.formats.json, [33](#)
- dazu.training_data.reader, [33](#)
- dazu.typing, [34](#)
- dazu.typing.message, [33](#)
- dazu.typing.model, [34](#)
- dazu.typing.module, [34](#)
- dazu.typing.training_data, [34](#)
- dazu.util, [28](#)
- dazu.utils, [35](#)
- dazu.utils.io, [34](#)
- dazu.version, [28](#)

A

Adapter (*class in dazu.adapters.adapter*), 28
as_dict() (*dazu.config.DazuConfig method*), 27

B

build() (*dazu.typing.message.Message class method*), 33
build_model_filename() (*in module dazu.components.engine*), 31

C

cache_key() (*dazu.components.component.Component class method*), 29
can_handle_language() (*dazu.components.component.Component class method*), 30
Component (*class in dazu.components.component*), 29
component_config_from_pipeline() (*in module dazu.config*), 27
component_names() (*dazu.config.DazuConfig property*), 27
components (*dazu.components.engine.Engine attribute*), 31
create() (*dazu.components.component.Component class method*), 30
create_directory() (*in module dazu.utils.io*), 34
create_directory_for_file() (*in module dazu.utils.io*), 34
create_path() (*in module dazu.utils.io*), 34
create_temporary_file() (*in module dazu.utils.io*), 34

D

dazu (*module*), 28
dazu.adapters (*module*), 29
dazu.adapters.adapter (*module*), 28
dazu.adapters.adapters (*module*), 29
dazu.adapters.adapters.google (*module*), 29
dazu.components (*module*), 31
dazu.components.component (*module*), 29
dazu.components.dialogue (*module*), 32

dazu.components.dialogue.simple (*module*), 31
dazu.components.engine (*module*), 31
dazu.components.nlu (*module*), 33
dazu.components.nlu.simple (*module*), 32
dazu.config (*module*), 27
dazu.constants (*module*), 28
dazu.registry (*module*), 28
dazu.training_data (*module*), 33
dazu.training_data.formats (*module*), 33
dazu.training_data.formats.json (*module*), 33
dazu.training_data.reader (*module*), 33
dazu.typing (*module*), 34
dazu.typing.message (*module*), 33
dazu.typing.model (*module*), 34
dazu.typing.module (*module*), 34
dazu.typing.training_data (*module*), 34
dazu.util (*module*), 28
dazu.utils (*module*), 35
dazu.utils.io (*module*), 34
dazu.version (*module*), 28
DazuConfig (*class in dazu.config*), 27
defaults (*dazu.components.component.Component attribute*), 30
dump_obj_as_json_to_file() (*in module dazu.utils.io*), 34

E

enable_async_loop_debugging() (*in module dazu.utils.io*), 34
Engine (*class in dazu.components.engine*), 31
evalCondition() (*in module dazu.components.dialogue.simple*), 32

F

fix_yaml_loader() (*in module dazu.utils.io*), 34
for_component() (*dazu.config.DazuConfig method*), 27

G

get() (*dazu.config.DazuConfig method*), 27

`get()` (*dazu.registry.Registry* class method), 28
`get()` (*dazu.typing.message.Message* method), 33
`get_dialog_anythinelse()` (in module *dazu.components.dialogue.simple*), 32
`get_dialog_welcome()` (in module *dazu.components.dialogue.simple*), 32
`getAdapter()` (*dazu.registry.Registry* class method), 28
`GoogleAdapter` (class in *dazu.adapters.adapters.google*), 29

I

`input()` (*dazu.adapters.adapter.Adapter* class method), 28
`input()` (*dazu.adapters.adapter.MessageAdapter* class method), 28
`input()` (*dazu.adapters.adapters.google.GoogleAdapter* class method), 29
`InvalidConfigError`, 27
`is_subdirectory()` (in module *dazu.utils.io*), 35
`items()` (*dazu.config.DazuConfig* method), 27

J

`json_to_string()` (in module *dazu.util*), 28
`JsonReader` (class in *dazu.training_data.formats.json*), 33

L

`language_list` (*dazu.components.component.Component* attribute), 30
`list_directory()` (in module *dazu.utils.io*), 35
`list_files()` (in module *dazu.utils.io*), 35
`list_subdirectories()` (in module *dazu.utils.io*), 35
`load()` (*dazu.components.component.Component* class method), 30
`load()` (*dazu.components.dialogue.simple.SimpleDialogue* class method), 31
`load()` (*dazu.components.nlu.simple.SimpleNLU* class method), 32
`load()` (*dazu.training_data.formats.json.JsonReader* class method), 33
`load()` (*dazu.training_data.reader.Reader* class method), 33
`load()` (in module *dazu.config*), 27

M

`Message` (class in *dazu.typing.message*), 33
`MessageAdapter` (class in *dazu.adapters.adapter*), 28
`Metadata` (class in *dazu.typing.model*), 34
`Model` (class in *dazu.typing.model*), 34
`Module` (class in *dazu.typing.module*), 34
`modules` (*dazu.registry.Registry* attribute), 28

N

`name()` (*dazu.adapters.adapter.MessageAdapter* class method), 28
`name()` (*dazu.adapters.adapters.google.GoogleAdapter* class method), 29
`name()` (*dazu.components.dialogue.simple.SimpleDialogue* class method), 31
`name()` (*dazu.components.nlu.simple.SimpleNLU* class method), 32
`name()` (*dazu.typing.module.Module* class method), 34

O

`output()` (*dazu.adapters.adapter.Adapter* class method), 28
`output()` (*dazu.adapters.adapter.MessageAdapter* class method), 28
`output()` (*dazu.adapters.adapters.google.GoogleAdapter* class method), 29
`override()` (*dazu.config.DazuConfig* method), 27
`override_defaults()` (in module *dazu.config*), 27

P

`partially_process()` (*dazu.components.component.Component* method), 30
`persist()` (*dazu.components.component.Component* method), 30
`persist()` (*dazu.components.dialogue.simple.SimpleDialogue* method), 31
`persist()` (*dazu.components.nlu.simple.SimpleNLU* method), 32
`pipeline` (*dazu.components.engine.Engine* attribute), 31
`pipeline_template()` (in module *dazu.registry*), 28
`prepare_partial_processing()` (*dazu.components.component.Component* method), 30
`process()` (*dazu.components.component.Component* method), 30
`process()` (*dazu.components.dialogue.simple.SimpleDialogue* method), 31
`process()` (*dazu.components.nlu.simple.SimpleNLU* method), 32
`provide_context()` (*dazu.components.component.Component* method), 30
`provides` (*dazu.components.component.Component* attribute), 30

R

`read_config_file()` (in module *dazu.utils.io*), 35
`read_file()` (in module *dazu.utils.io*), 35
`read_json_file()` (in module *dazu.utils.io*), 35

[read_yaml\(\)](#) (in module *dazu.utils.io*), 35
[read_yaml_file\(\)](#) (in module *dazu.utils.io*), 35
[Reader](#) (class in *dazu.training_data.reader*), 33
[Registry](#) (class in *dazu.registry*), 28
[registry\(\)](#) (*dazu.registry.Registry* class method), 28
[registryAdapter\(\)](#) (*dazu.registry.Registry* class method), 28
[replace_environment_variables\(\)](#) (in module *dazu.utils.io*), 35
[required_packages\(\)](#) (*dazu.components.component.Component* class method), 30
[requires](#) (*dazu.components.component.Component* attribute), 30
[respond\(\)](#) (*dazu.components.engine.Engine* method), 31

S

[set\(\)](#) (*dazu.typing.message.Message* method), 33
[set_component_attr\(\)](#) (*dazu.config.DazuConfig* method), 27
[similarity\(\)](#) (in module *dazu.components.nlu.simple*), 32
[SimpleDialogue](#) (class in *dazu.components.dialogue.simple*), 31
[SimpleNLU](#) (class in *dazu.components.nlu.simple*), 32

T

[tokenize\(\)](#) (in module *dazu.components.nlu.simple*), 33
[train\(\)](#) (*dazu.components.component.Component* method), 30
[train\(\)](#) (*dazu.components.dialogue.simple.SimpleDialogue* method), 32
[train\(\)](#) (*dazu.components.engine.Engine* method), 31
[train\(\)](#) (*dazu.components.nlu.simple.SimpleNLU* method), 32
[TrainingData](#) (class in *dazu.typing.training_data*), 34

U

[unarchive\(\)](#) (in module *dazu.utils.io*), 35
[UnsupportedLanguageError](#), 31

V

[validate_data\(\)](#) (*dazu.adapters.adapter.Adapter* class method), 28
[validate_data\(\)](#) (*dazu.adapters.adapter.MessageAdapter* class method), 28
[validate_data\(\)](#) (*dazu.adapters.adapters.google.GoogleAdapter* class method), 29
[validate_data\(\)](#) (*dazu.training_data.reader.Reader* class method), 33
[view\(\)](#) (*dazu.config.DazuConfig* method), 27

W

[write_text_file\(\)](#) (in module *dazu.utils.io*), 35
[write_yaml_file\(\)](#) (in module *dazu.utils.io*), 35

Z

[zip_folder\(\)](#) (in module *dazu.utils.io*), 35