# Data Stack

*Release 1.0*

**Eric Daoud**

**Oct 26, 2020**

# CONTENTS:


**1 1. Presentation** **3**

**2 2. Installation** **5**

**3 3. Usage** **7**
- 3.1 3.1 Launch the Docker stack . . . 7
- 3.2 3.2 Unit testing . . . 7
- 3.3 3.3 Generating the Sphinx docs . . . 7

**4 4. References** **9**
- 4.1 src . . . 9
- 4.2 API Reference . . . 11

**5 Indices and tables** **17**

**Python Module Index** **19**

**Index** **21**

CHAPTER

ONE

# 1. PRESENTATION

A sample data stack running on Docker, that contains the following components:

- Airflow
- Metabase
- MariaDB, with PHPMyAdmin
- Postgres, with PHPPgAdmin
- Doccano data labelling interface
- Nginx as reverse proxy
- Sphinx auto-generated documentation
- A template python module, usable in Airflow DAGS
- A template machine learning package, using Pytorch
- A `ml_helper` package, that provides functions to store machine learning models results and parameters in a database.
- A `utils` package with utilities functions.
- Unit-testing with pytest library

CHAPTER

TWO

# 2. INSTALLATION

You will need to have the following software installed:

- python3
- virtualenv
- Docker
- docker-compose

Once you're good, create a virtual environment in install the pre-requisite python libraries:

```
virtualenv venv;
source venv/bin/activate;
pip install -r requirements.txt;
```

CHAPTER

THREE

# 3. USAGE

## 3.1 3.1 Launch the Docker stack

Run it with:

```
docker-compose up -d
```

Then visit:

- localhost:3000: for Metabase
- localhost:8080: for Airflow
- localhost:8000: for Doccano

Add your Airflow DAGS in the dags folder.

## 3.2 3.2 Unit testing

Run the unit tests with:

```
pytest tests
```

## 3.3 3.3 Generating the Sphinx docs

Generate the Sphinx documentation with:

```
sphinx-apidoc ./src -o docs/source -M;
cd docs && make html && open build/html/index.html;
```

CHAPTER

FOUR

# 4. REFERENCES

- puckel/docker-airflow
- chakki-works/doccano

## 4.1 src

### 4.1.1 src package

#### Subpackages

#### src.ml package

#### Submodules

#### src.ml.pytorch_example module

From: https://pytorch.org/tutorials/beginner/text_sentiment_ngrams_tutorial.html

src.ml.pytorch_example.**BATCH_SIZE = 16**

**class** src.ml.pytorch_example.**TextSentiment**(*vocab_size*, *embed_dim*, *num_class*)

Bases: torch.nn.modules.module.Module

**forward**(*text*, *offsets*)

**init_weights**()

src.ml.pytorch_example.**device = device(type='cpu')**

src.ml.pytorch_example.**generate_batch**(*batch*)

src.ml.pytorch_example.**main**()

src.ml.pytorch_example.**test**(*model*, *criterion*, *data_*)

src.ml.pytorch_example.**train_func**(*model*, *optimizer*, *criterion*, *scheduler*, *sub_train_*)

#### src.ml_helper package

#### Submodules

### src.ml_helper.model module

**class** src.ml_helper.model.**Base**(***kwargs*)
Bases: object

The most base type

**metadata = MetaData(bind=None)**

src.ml_helper.model.**ENGINE_URL = 'postgres://user:password@localhost:5432/ml_helper'**

**class** src.ml_helper.model.**Epoch**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

**created_at**

**eval_F1**

**eval_acc**

**eval_loss**

**model_id**

**number**

**training_F1**

**training_acc**

**training_loss**

**uuid**

**class** src.ml_helper.model.**Model**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

**epoch**

**id**

**last_updated**

**params**

**uuid**

src.ml_helper.model.**Session = sessionmaker(class_='Session', bind=Engine(postgres://user:***

src.ml_helper.model.**engine = Engine(postgres://user:***@localhost:5432/ml_helper)**

src.ml_helper.model.**init_db**()

src.ml_helper.model.**metadata = MetaData(bind=None)**

### src.ml_helper.training module

src.ml_helper.training.**delete_model**(*model_id*)

src.ml_helper.training.**hash_parameters**(*params*)

src.ml_helper.training.**register_epoch_in_db**(*model_id*, *epoch_number*, ***kwargs*)

src.ml_helper.training.**register_model_in_db**(*model_id*, *params*)

src.ml_helper.training.**retrieve_best_model_params**()

#### Submodules

#### src.example_module module

A python example module.

src.example_module.**example_function**()
Basic function that returns a string

## 4.2 API Reference

This page contains auto-generated API reference documentation[1].

### 4.2.1 src

#### Subpackages

**src.ml**

#### Submodules

**src.ml.pytorch_example**

From: https://pytorch.org/tutorials/beginner/text_sentiment_ngrams_tutorial.html

#### Module Contents

#### Classes

| | |
|---|---|
| *TextSentiment* | Base class for all neural network modules. |

#### Functions

| | |
|---|---|
| *generate_batch*(batch) | |
| *train_func*(model, optimizer, criterion, scheduler, sub_train_) | |
| *test*(model, criterion, data_) | |
| *main*() | |

src.ml.pytorch_example.**BATCH_SIZE = 16**

src.ml.pytorch_example.**device**

**class** src.ml.pytorch_example.**TextSentiment**(*vocab_size*, *embed_dim*, *num_class*)
Bases: torch.nn.Module

Base class for all neural network modules.

[1] Created with sphinx-autoapi

Your models should also subclass this class.

Modules can also contain other Modules, allowing to nest them in a tree structure. You can assign the submodules as regular attributes:

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return F.relu(self.conv2(x))
```

Submodules assigned in this way will be registered, and will have their parameters converted too when you call `to()`, etc.

**init_weights**(*self*)

**forward**(*self*, *text*, *offsets*)

src.ml.pytorch_example.**generate_batch**(*batch*)

src.ml.pytorch_example.**train_func**(*model*, *optimizer*, *criterion*, *scheduler*, *sub_train_*)

src.ml.pytorch_example.**test**(*model*, *criterion*, *data_*)

src.ml.pytorch_example.**main**()

### src.ml_helper

#### Submodules

### src.ml_helper.model

#### Module Contents

#### Classes

| |
|---|
| *Model* |
| *Epoch* |

#### Functions

| |
|---|
| *init_db*() |

src.ml_helper.model.**ENGINE_URL = postgres://user:password@localhost:5432/ml_helper**

src.ml_helper.model.**metadata**

src.ml_helper.model.**Base**

**class** src.ml_helper.model.**Model**
Bases: *Base*

**__tablename__ = model**

**uuid**

**id**

**last_updated**

**params**

**epoch**

**class** src.ml_helper.model.**Epoch**
Bases: *Base*

**__tablename__ = epoch**

**__table_args__**

**uuid**

**model_id**

**created_at**

**number**

**training_loss**

**eval_loss**

**training_F1**

**eval_F1**

**training_acc**

**eval_acc**

src.ml_helper.model.**engine**

src.ml_helper.model.**Session**

src.ml_helper.model.**init_db**()

### src.ml_helper.training

#### Module Contents

#### Functions

| | |
|---|---|
| *_commit_object*(obj) | |
| *register_model_in_db*(model_id, params) | |
| *register_epoch_in_db*(model_id, epoch_number, **kwargs) | |
| *retrieve_best_model_params*() | |
| *hash_parameters*(params) | |
| *delete_model*(model_id) | |

src.ml_helper.training.**_commit_object**(*obj*)

src.ml_helper.training.**register_model_in_db**(*model_id*, *params*)

src.ml_helper.training.**register_epoch_in_db**(*model_id*, *epoch_number*, ***kwargs*)

src.ml_helper.training.**retrieve_best_model_params**()

src.ml_helper.training.**hash_parameters**(*params*)

src.ml_helper.training.**delete_model**(*model_id*)

### src.utils

#### Submodules

### src.utils.io

#### Module Contents

#### Functions

| | |
|---|---|
| load_pretrained_embeddings(embeddings_file, embeddings_dim, word_to_ix, skip_header=False) | Load pretrained embeddings weights. |

src.utils.io.**load_pretrained_embeddings**(*embeddings_file*, *embeddings_dim*, *word_to_ix*, *skip_header=False*)

Load pretrained embeddings weights. For the words that don't have a pre-trained embedding, we assign them a randomly initialized one. :param embeddings_file: Weights file :type embeddings_file: str :param embeddings_dim: Embeddings dim :type embeddings_dim: int :param word_to_ix: Word to index mapper :type word_to_ix: dict

**Returns** pre-trained embeddings matrix

**Return type** np.matrix

### src.utils.text

#### Module Contents

#### Functions

| | |
|---|---|
| isolate_punctuation(text) | Isolate punctuation in a sentence. |
| replace_urls(text, replace_with='<URL>') | Replace urls in a sentence with a chosen string. |

src.utils.text.**isolate_punctuation**(*text*)

Isolate punctuation in a sentence.

```
>>> split_punctuation('Hi there!')
'Hi there !'
```

**Parameters** **text** (*str*) – Input sentence

**Returns** Output sentence with isolated punctuation

**Return type** str

src.utils.text.**replace_urls**(*text*, *replace_with='<URL>'*)

Replace urls in a sentence with a chosen string.

```
>>> replace_urls("I love https://github.com")
"I love <URL>"
```

**Parameters**

- **text** (*str*) – Input sentence
- **replace_with** (*str, optional*) – string to replace the url with. Defaults to "<URL>".

**Returns** Output sentence with replaced url

**Return type** str

### src.utils.vocabulary

#### Module Contents

#### Functions

| | |
|---|---|
| make_char_to_ix() | Make a character to index dictionary. |
| make_word_to_ix(train_sentences, char_to_split_at=' ', unk_tag='<UNK>') | Make a word to index dictionary |

src.utils.vocabulary.**make_char_to_ix**()

Make a character to index dictionary.

**Returns** character to index

**Return type** dict

src.utils.vocabulary.**make_word_to_ix**(*train_sentences*, *char_to_split_at=' '*, *unk_tag='<UNK>'*)

Make a word to index dictionary

**Parameters**

- **train_sentences** (*list*) – list of sentences
- **char_to_split_at** (*str, optional*) – str. Character to use to split the sentence (for tokenization). Defaults to " ".
- **unk_tag** (*str, optional*) – Unknown tag. Defaults to "<UNK>".

**Returns** [description]

**Return type** [type]

### Submodules

#### src.example_module

A python example module.

### Module Contents

### Functions

| | |
|---|---|
| *example_function*() | Basic function that returns a string |

src.example_module.**example_function**()
Basic function that returns a string

CHAPTER

FIVE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## S

src, 9
src.example_module, 11
src.ml, 9
src.ml.pytorch_example, 9
src.ml_helper, 9
src.ml_helper.model, 10
src.ml_helper.training, 10

# INDEX

## Symbols


__table_args__ (*src.ml_helper.model.Epoch attribute*), 13
__tablename__ (*src.ml_helper.model.Epoch attribute*), 13
__tablename__ (*src.ml_helper.model.Model attribute*), 13
_commit_object() (*in module src.ml_helper.training*), 14


## B


Base (*class in src.ml_helper.model*), 10
Base (*in module src.ml_helper.model*), 12
BATCH_SIZE (*in module src.ml.pytorch_example*), 9, 11


## C


created_at (*src.ml_helper.model.Epoch attribute*), 10, 13


## D


delete_model() (*in module src.ml_helper.training*), 10, 14
device (*in module src.ml.pytorch_example*), 9, 11


## E


engine (*in module src.ml_helper.model*), 10, 13
ENGINE_URL (*in module src.ml_helper.model*), 10, 12
Epoch (*class in src.ml_helper.model*), 10, 13
epoch (*src.ml_helper.model.Model attribute*), 10, 13
eval_acc (*src.ml_helper.model.Epoch attribute*), 10, 13
eval_F1 (*src.ml_helper.model.Epoch attribute*), 10, 13
eval_loss (*src.ml_helper.model.Epoch attribute*), 10, 13
example_function() (*in module src.example_module*), 11, 16


## F


forward() (*src.ml.pytorch_example.TextSentiment method*), 9, 12


## G


generate_batch() (*in module src.ml.pytorch_example*), 9, 12


## H


hash_parameters() (*in module src.ml_helper.training*), 10, 14


## I


id (*src.ml_helper.model.Model attribute*), 10, 13
init_db() (*in module src.ml_helper.model*), 10, 13
init_weights() (*src.ml.pytorch_example.TextSentiment method*), 9, 12
isolate_punctuation() (*in module src.utils.text*), 14


## L


last_updated (*src.ml_helper.model.Model attribute*), 10, 13
load_pretrained_embeddings() (*in module src.utils.io*), 14


## M


main() (*in module src.ml.pytorch_example*), 9, 12
make_char_to_ix() (*in module src.utils.vocabulary*), 15
make_word_to_ix() (*in module src.utils.vocabulary*), 15
metadata (*in module src.ml_helper.model*), 10, 12
metadata (*src.ml_helper.model.Base attribute*), 10
Model (*class in src.ml_helper.model*), 10, 12
model_id (*src.ml_helper.model.Epoch attribute*), 10, 13


## N


number (*src.ml_helper.model.Epoch attribute*), 10, 13


## P


params (*src.ml_helper.model.Model attribute*), 10, 13

## R

register_epoch_in_db() (*in module src.ml_helper.training*), 10, 14
register_model_in_db() (*in module src.ml_helper.training*), 10, 14
replace_urls() (*in module src.utils.text*), 15
retrieve_best_model_params() (*in module src.ml_helper.training*), 10, 14

## S

Session (*in module src.ml_helper.model*), 10, 13
src (*module*), 9, 11
src.example_module (*module*), 11, 16
src.ml (*module*), 9, 11
src.ml.pytorch_example (*module*), 9, 11
src.ml_helper (*module*), 9, 12
src.ml_helper.model (*module*), 10, 12
src.ml_helper.training (*module*), 10, 13
src.utils (*module*), 14
src.utils.io (*module*), 14
src.utils.text (*module*), 14
src.utils.vocabulary (*module*), 15

## T

test() (*in module src.ml.pytorch_example*), 9, 12
TextSentiment (*class in src.ml.pytorch_example*), 9, 11
train_func() (*in module src.ml.pytorch_example*), 9, 12
training_acc (*src.ml_helper.model.Epoch attribute*), 10, 13
training_F1 (*src.ml_helper.model.Epoch attribute*), 10, 13
training_loss (*src.ml_helper.model.Epoch attribute*), 10, 13

## U

uuid (*src.ml_helper.model.Epoch attribute*), 10, 13
uuid (*src.ml_helper.model.Model attribute*), 10, 13