

SEMINAR

CONFIGURATION AND RESULTS MANAGEMENT WITH MLFLOW, HYDRA AND POUTYNE

JANUARY 19, 2021

OBJECTIVES OF THE PRESENTATION

- Introduce configuration and results management tools.
- Developing good practices.
- Improve your productivity.

VOTRE CONFÉRENCIER



DAVID BEAUCHEMIN

Ph.D. candidate

Department of Computer Science and Software Engineering

- Introduced to reproducible research in 2016 (R Markdown et **git**)
- Participation in REPROLANG of the LREC conference [Garneau et al., 2020]
- Active member in the development of a library to facilitate reproducibility (**Poutyne** ↗*)

ON THE MENU



Configuration management



Results management

The Management of a Project

MANAGEMENT OF CONFIGURATION PARAMETERS

```
001    @experiment.config
002    def config() :
003        seed = 42
004        num_runs = 10
005        iteration = 0
006        source_language = "en"
007        target_language = "de"
008        src_input = "path" # The input source embeddings
009        trg_input = "2e path" # The input target embeddings
010        other_input = "3e path" # Commentaire pas clair
    :
395    n-th parameters
```

MANAGEMENT OF CONFIGURATION PARAMETERS

Which one does that
again?

MANAGEMENT OF CONFIGURATION PARAMETERS

Which one does that
again?

Which ones
necessarily go
together?

MANAGEMENT OF CONFIGURATION PARAMETERS

Which one does that
again?

Which ones
necessarily go
together?

Which ones are really
essential?

MANAGEMENT OF CONFIGURATION PARAMETERS

Which one does that
again?

Which ones
necessarily go
together?

Which ones are really
essential?

How to organize
them?

RESULTS MANAGEMENT

```
|_ res_1.txt  
|_ res_2.txt  
|_ res_3.txt  
|_ res_4.txt  
|_ res_5_good.txt  
|_ res_5.txt  
|_ res_6_fix_a.txt  
|_ :  
|_ n-th results file
```

RESULTS MANAGEMENT

Which configuration
(already) used?

RESULTS MANAGEMENT

Which configuration
(already) used?

Success or failure?

RESULTS MANAGEMENT

Which configuration
(already) used?

Success or failure?

Which one is the
best?

RESULTS MANAGEMENT

Which configuration
(already) used?

Success or failure?

Which one is the
best?

How to organize
them?

The Solutions

ON THE MENU



Configuration management



Results management

CONFIGURATION MANAGEMENT



Simple and efficient

CONFIGURATION MANAGEMENT



Simple and efficient



Facilitates experimentation

CONFIGURATION MANAGEMENT



Simple and efficient



Facilitates experimentation



Scalable

POSSIBLE SOLUTIONS

- Arguments (e.g. argparse, configparser)
- Text file
- JSON
- YAML
- ...

HYDRA 



Open source and MIT
license



YAML structured
configuration files



Hierarchical
configuration files



Configurations
sweeper

STRUCTURED CONFIGURATION

```
data_loader :  
  batch_size : 2048 # the batch size  
setting :  
  seed : 42  
  device : "cuda:0"  
defaults :  
  - optimizer : SGD  
  - model : bi_lstm  
  - dataset : canadian  
  - embeddings : fast_text  
trainer :  
  num_epochs : 1  
  patience : 30
```

HIERARCHICAL CONFIGURATION

```
conf
├── config.yaml
├── dataset
│   ├── canadian.yaml
│   └── netherlands.yaml
├── embeddings
│   └── fast_text.yaml
├── model
│   ├── bi_lstm_bidirectionnal.yaml
│   ├── bi_lstm.yaml
│   ├── lstm_bidirectionnal.yaml
│   └── lstm.yaml
├── optimizer
│   ├── adam.yaml
│   └── SGD.yaml
```


HIERARCHICAL CONFIGURATION

optimizer : SGD

```
optimizer :  
  lr : 0.1  
  type : sgd
```

EXAMPLE

```
@hydra.main(config_path='conf/config.yaml')  
def main(cfg) :  
    lr = cfg.optimizer.lr #0.1
```

CONFIGURATIONS SWEEPER

```
python main.py -multirun task=1,2,3,4,5
```

```
python main.py -m 'main.x=int(interval(-5, 5))' 'main.y=int(interval(-5, 10))'
```

BONUS!

- Automatic and customizable logging
- Parametric instantiation

```
model :  
  _target_ : models.LSTMNetwork  
  hidden_state_dim : 300  
  num_hidden_layer : 2  
  dropout : 0.4
```

EXAMPLE

```
log = logging.getLogger(__name__)
@hydra.main(config_path='conf/config.yaml')
def main(cfg) :
    log.info("Init of the training")
    :
    network = instantiate(cfg.model)
```

NEGATIVE POINT

```
hydra.utils.get_original_cwd()
```

ON THE MENU



Configuration management



Results management

RESULTS MANAGEMENT



Simple to use

RESULTS MANAGEMENT



Simple to use



Experimental logging

RESULTS MANAGEMENT



Simple to use



Experimental logging



Quick visualization of
experiments

MLFLOW TRACKING *



Open source and
Apache 2.0 license



Automatic logging



Simple visualization



Integration with
Poutyne

AUTOMATIC LOGGING

- Code version (**git**)*
- Training timestamp
- Training success/failure
- Computer configuration
- User

SIMPLE VISUALIZATION

```
mlflow server -p 5000 -h 127.0.0.1 --backend-store-uri file:///absolute/path
```

SIMPLE VISUALIZATION

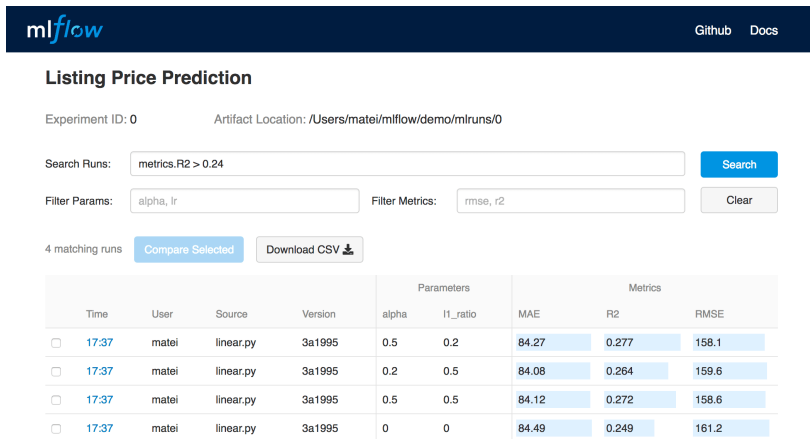


Figure 1 – Introducing MLflow : an Open Source Machine Learning Platform *

SIMPLE VISUALIZATION

- Sorting on experiments
- Research of experiments
- Queries on results
- Export of results
- Visualization of metrics

INTEGRATION WITH **POUTYNE** *

The "basic" version involves manual logging

- configuration parameters,
- metrics at each step and iteration,
- the version of the code.

INTÉGRATION AVEC **POUTYNE** *

The solution, [MLFlowWriter](#), a callback allowing to journalize

- semi-automatically the configuration parameters,
- automatically the metrics at each step and iteration,
- automatically the version of the code,
- manually a model,
- automatically test metrics during a test phase.

EXAMPLE

```
@hydra.main(config_path='conf/config.yaml')
def main(cfg) :
    :
    mlflow_logger = MLFlowLogger(experiment_name="experiment")
    mlflow_logger.log_config_params(config_params=cfg)
    :
    mlflow_logger.log_model()
```

NEGATIVE POINT

The documentation is not always easy to navigate.

What's next?

PRÉSENTATION DES RÉSULTATS



Automatic generation of tables





Dynamic report



Iterations of experiments

TO GO FURTHER (IN ORDER)

- Training status notification **Notif** *
- **Continuous Machine Learning (CML)** *


QUESTIONS



SEMINAR

**THANK YOU FOR
LISTENING!**

REFERENCES i

-  Garneau, N., Godbout, M., Beauchemin, D., Durand, A., and Lamontagne, L. (2020).
A Robust Self-Learning Method for Fully Unsupervised Cross-Lingual Mappings of Word Embeddings : Making the Method Robustly Reproducible as Well.