

WEBINAIRE

GESTION DE LA CONFIGURATION ET DES RÉSULTATS AVEC MLFLOW/ WEIGHTS AND BIASES, HYDRA ET POUTYNE

3 MAI 2022

OBJECTIFS DE LA PRÉSENTATION

- Initier aux outils de gestion de la configuration et des résultats.
- Développer de bonnes pratiques.
- Améliorer votre productivité.


VOTRE CONFÉRENCIER



DAVID BEAUCHEMIN

Candidat au doctorat

Département d'informatique et de génie logiciel

- Introduit à la recherche reproductible en 2016 (R Markdown et **git**)
- Participation à REPROLANG de la conférence LREC [Garneau et al., 2020]
- Membre actif dans le développement d'une librairie facilitant la reproductibilité (**Poutyne** *)

AU MENU



Gestion de la configuration



Gestion des résultats

La gestion d'un projet

GESTION DES PARAMÈTRES DE CONFIGURATION

```
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    ne paramètre
```

GESTION DES PARAMÈTRES DE CONFIGURATION

Lequel fait ça déjà ?

GESTION DES PARAMÈTRES DE CONFIGURATION

Lequel fait ça déjà ?

Lesquels vont
nécessairement
ensemble ?

GESTION DES PARAMÈTRES DE CONFIGURATION

Lequel fait ça déjà ?

Lesquels vont
nécessairement
ensemble ?

Lesquels sont
vraiment essentiels ?

GESTION DES PARAMÈTRES DE CONFIGURATION

Lequel fait ça déjà ?

Lesquels vont
nécessairement
ensemble ?

Lesquels sont
vraiment essentiels ?

Comment les
organiser ?

GESTION DES RÉSULTATS

- 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^{e} fichier de résultats

GESTION DES RÉSULTATS

Quelle configuration
(déjà) utilisée ?

GESTION DES RÉSULTATS

Quelle configuration
(déjà) utilisée?

Succès ou échec?

GESTION DES RÉSULTATS

Quelle configuration
(déjà) utilisée?

Succès ou échec?

Lequel est le
meilleur?

GESTION DES RÉSULTATS

Quelle configuration
(déjà) utilisée?

Succès ou échec?

Lequel est le
meilleur?

Comment les
organiser?

Les solutions

AU MENU



Gestion de la configuration



Gestion des résultats

GESTION DE LA CONFIGURATION



Simple et efficace

GESTION DE LA CONFIGURATION



Simple et efficace



Facilite l'expérimentation

GESTION DE LA CONFIGURATION



Simple et efficace



Facilite l'expérimentation



Extensible

SOLUTIONS POSSIBLE

- Arguments (e.g. argparse, configparser)
- Fichier texte
- JSON
- YAML
- ...

HYDRA 



*Open source et
licence MIT*



Fichiers de
configurations
structurés YAML



Fichiers de
configurations
hiérarchiques



Balayage de
configurations

CONFIGURATION STRUCTURÉ

data_loader :

batch_size : 2048

setting :

seed : 42

device : "cuda:0"

defaults :

- optimizer : SGD

- model : bi_lstm

- dataset : canadian

- embeddings : fast_text

trainer :

num_epochs : 1

patience : 30

CONFIGURATION HIÉRARCHIQUE

```
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
```


CONFIGURATION HIÉRARCHIQUE

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
```

BALAYAGE DE CONFIGURATIONS

```
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))'
```

EN BONUS

- Journalisation automatique et personnalisable
- Instanciation paramétrique

```
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)
```

POINT NÉGATIF

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

AU MENU



Gestion de la configuration



Gestion des résultats

GESTION DES RÉSULTATS



Simple à utiliser

GESTION DES RÉSULTATS



Simple à utiliser



Journalisation des
expérimentations

GESTION DES RÉSULTATS



Simple à utiliser



Journalisation des
expérimentations



Visualisation rapide des
expérimentations

MLFLOW TRACKING *



Open source et
licence Apache 2.0



Journalisation
automatique



Visualisation simple



Intégration avec
Poutyne

JOURNALISATION AUTOMATIQUE

- Version du code (**git**)*
- Horodatage de l'entraînement
- Succès/échec de l'entraînement
- Configuration de l'ordinateur
- Utilisateur

VISUALISATION SIMPLE

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

VISUALISATION SIMPLE

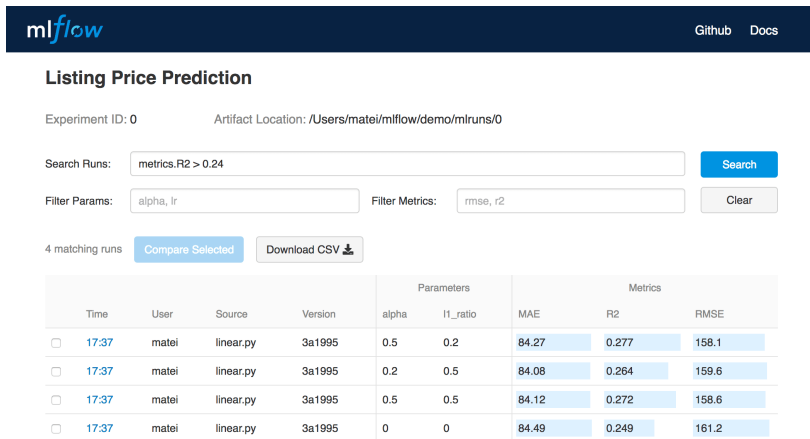


Figure 1 : Introducing MLflow : an Open Source Machine Learning Platform 

VISUALISATION SIMPLE

- Tri sur les expérimentations
- Recherche des expérimentations
- Requêtes sur les résultats
- Exportation des résultats
- Visualisation des métriques

INTÉGRATION AVEC **POUTYNE** *

La version de « base » implique de journaliser manuellement

- les paramètres de configuration,
- les métriques à chaque étape et itération,
- la version du code.

INTÉGRATION AVEC POUTYNE *

La solution, [MLFlowLogger](#), un *callback* permettant de journaliser

- semi-automatiquement les paramètres de configuration,
- automatiquement les métriques à chaque étape et itération,
- automatiquement la version du code,
- automatiquement les métriques de test lors d'une phase de test.

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)
```

POINT NÉGATIF

- La documentation n'est pas toujours facile à naviguer, et
- très complexe de journaliser des artefacts, nous n'avons pas réussi.

WEIGHTS & BIASES *



Journalisation
automatique



Visualisation simple
et puissante




Génération de
graphique
automatique



Intégration avec
Poutyne

JOURNALISATION AUTOMATIQUE

- Version du code (**git**)
- Horodatage de l'entraînement
- Succès/échec de l'entraînement
- Configuration de l'ordinateur
- Utilisateur
- Génération automatique de graphique
- Journalisation des artefacts et *sweeps*
- Via un Saas ou **client** *

VISUALISATION SIMPLE

```
wandb local
```

VISUALISATION SIMPLE

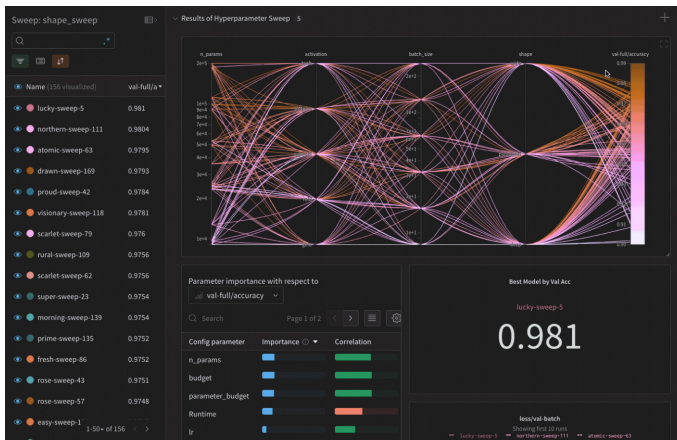


Figure 2 : Demo *

VISUALISATION SIMPLE

- Tri sur les expérimentations
- Recherche des expérimentations
- Requêtes sur les résultats
- Exportation des résultats
- Visualisation des métriques
- Interface avec **beaucoup** d'informations

INTÉGRATION AVEC **POUTYNE** *

La version de « base » implique de journaliser manuellement

- les paramètres de configuration,
- les métriques à chaque étape et itération,
- la version du code,
- les artefacts.

INTÉGRATION AVEC POUTYNE *

La solution, [WandbLogger](#), un *callback* permettant de journaliser

- semi-automatiquement les paramètres de configuration,
- automatiquement les métriques à chaque étape et itération,
- automatiquement la version du code,
- **automatiquement un modèle**,
- automatiquement les métriques de test lors d'une phase de test, et
- manuellement des images et figures.

EXAMPLE

```
@hydra.main(config_path='conf/config.yaml')
def main(cfg) :
    :
    wandb_logger = WandB(name="experiment", project="A project")
    wandb_logger.log_config_params(config_params=cfg)
    :
    image = wandb.Image('a/image.png', caption="a caption")
    wandb_logger.run.log("My image" : image)
```

POINT NÉGATIF

Solution propriétaire (gratuit académique 40\$/mois utilisateur sinon).

La suite

PRÉSENTATION DES RÉSULTATS



Génération automatique des tableaux



Rapport dynamique



Itérations d'expérimentations

POUR ALLER PLUS LOIN (EN ORDRE)

- Notification de l'état d'entraînement **Notif** *
- **Continuous Machine Learning (CML)** *


PÉRIODE DE QUESTIONS



WEBINAIRE

**MERCI DE VOTRE
ÉCOUTE!**

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.**