Weekly P-AMI<Q>

2025/05/07 GesonAnko

自己紹介

- げそん (GesonAnko)

  X(Twitter)@GesonAnkoVR

- ML集会 主催
  - 自律機械知能の研究開発
  - Pythonで機械学習のツール作る（機械学習より得意かもしれない）

自律機械知能に脳みそを焼かれた人。

今週の進捗

- ゴールデンウィークなのであんまり無し
- Hydra YAML言語とその言語サーバを作った話

Hydra YAML言語とは

Hydra YAML言語とは

の前に

Hydraとは

- Pythonの設定管理ツール
  - **Hydraは、研究やその他の複雑なアプリケーションの開発を簡素化するオープンソースのPythonフレームワークです。**

- 主な機能
  - **複数の設定ファイルから階層的に構成**
  - コマンドラインから上書き（追加もできる）
    - コマンドラインの補完もある
  - 複数のジョブを一つのコマンドで実行したり…

参考: https://hydra.cc/docs/intro/

複数の設定ファイルで階層構成

- 分割して記述、結合。
  - 最上位の設定ファイルで名称を選択し、統合できる。

```
model/
   resnet.yml
   densenet.yml
   lenet.yml

data/
   mnist.yml
   cifer10.yml

train.yml
```

train.yamlの中身

```
defaults:
  - model: resnet  ← resnet.ymlを参照
  - data: mnist  ← mnist.ymlを参照
```

train.yamlが読み込まれた結果

```
model: ...（resnet.ymlの中身）
data: ...（mnist.ymlの中身）
```

ML Shukai

見るからにとても強力✨

Hydra YAML言語とは

- Hydraで扱われる設定ファイルの記述言語

- どんな機能があるのか
  - **Pythonオブジェクトのインスタンス化**
    - `_target_: path.to.class`
      `arg: …`
  - （これは OmegaConfの機能だが…）
    - 他の場所の設定値の参照 （相対・絶対パスどちらも）
      - “`key: ${path.to.value}`”
    - 関数マクロ呼び出し
      - “`key: ${func:some_arg}`”

Hydra YAML言語とは

- Hydraで扱われる設定ファイルの記述言語

- どんな機能があるのか
  - **Pythonオブジェクトのインスタンス化**
    - `_target_: path.to.class`
      `arg: …`
  - （これは OmegaConfの機能だが…）
    - 他の場所の設定値の参照 （相対・絶対パスどちらも）
      - “`key: ${path.to.value}`”
    - 関数マクロ呼び出し
      - “`key: ${func:some_arg}`”

なんか
やばくなってきた😅

オブジェクトのインスタンス化

- **“_target_”** キーにPythonのimportパスを指定
  - “hydra.utils.instantiate” に渡す。
    “arg1” と “arg2” が クラスのコンストラクタに渡される。

```
object:↓
  _target_: sample_python_project.YourClass
  arg1: 1↓
  arg2: "aaa"↓
```

  - “_args_” キーで位置引数として渡すことも可能

```
object:↓
  _target_: sample_python_project.YourClass
  _args_: ↓
  - 0↓
  - 1↓
```

他の場所の設定値の参照

- **“${}”** で囲んだら補完呼び出し

```
reference:
  value1: 100
  value2: ${.value1}
  deep:
    value3: ${reference.value1}
    value4: ${..value2}
```

- “value2” は “value1” を相対参照する → 100
- “value3” も “value1” を絶対参照する → 100
- “value4” は “value2” を相対参照する → 100 (連鎖参照)

他の場所の設定値の参照

- “${}” で囲んだら補完呼び出し

```
reference:
  value1: 100
  value2: ${.value1}
  deep:
    value3: ${reference.value1}
    value4: ${..value2}
```

これに設定ファイルの
階層的構成が入ったら…

- “value2” は “value1” を相対参照する → 100
- “value3” も “value1” を絶対参照する → 100
- “value4” は “value2” を相対参照する → 100 (連鎖参照)

関数マクロの呼び出し

- **“${func:}”** で登録済みの関数を呼び出す。

```
result: ${sum:1,2,3}
```

  - “result” は “sum” 関数に 1, 2, 3 の引数を与えた結果。

  - もちろん、参照もできる😊

```
result: ${sum:1,2,${value}}
```

これはYAML言語？

これはYAML言語？

NO!

Hydra YAML言語✨

しかし、エディタ拡張は無い。

```
i_jepa_predictor:↓
  _target_: ami.models.model_wrapper.ModelWrapper↓
  default_device: ${devices.0}↓
  has_inference: False↓
  model:↓
    _target_: ami.models.i_jepa.IJEPAPredictor↓
    n_patches:↓
      - ${python.eval:"${shared.image_height} // ${models.i_jepa_target_encoder.model.patch_size}"}↓
      - ${python.eval:"${shared.image_width} // ${models.i_jepa_target_encoder.model.patch_size}"}↓
    context_encoder_embed_dim: ${models.i_jepa_target_encoder.model.embed_dim}↓
    predictor_embed_dim: 384↓
    depth: 6↓
    num_heads: 12      You, 28 秒前 • Uncommitted changes
```

Hydra YAMLの言語サーバ作ってます！

ML Shukai

- 主な機能
  - **構文ハイライト**
  - **補完**
    - 特殊キー, 関数
    - Pythonのimportパス
    - 値参照 (未実装)
  - **リント（未実装）**
    - 無効な参照、循環参照
    - importパスの検証
  - **定義・参照元のへの移動（未実装）**

```yaml
# @package _global_
#
# Normal Test Cases
sample_project:
  _target_: sample_python_project.YourClass
  _convert_: none
  _recursive_: true
  _partial_: true
  arg1: 784
  arg2: 10
#
class_method:
  _target_: sample_python_project.YourClass.cls_method
  arg: 0
#
hydra:
  _target_: hydra.utils.get_method
  path: sample_python_project.hello
#
# Nested references.
nested:
  value1: 100
  value2: ${.value1}
  deep:
    value3: ${..value1}
    value4: ${.value3}
#
python_eval: ${python.eval:${value},${value2}}
```

作る機能はたくさんあるよ