Python: define & class

Wonkeun J.

Define

$$y = f(x) \Rightarrow y_1, y_2 = f(x_1, x_2, \cdots, x_3)$$

Example 01.

```
def square(x):
    return x ** 2

ans = square(5)
```

Example 02.

```
import numpy as np

def multiply_and_add(*x):
    return np.prod(x), sum(x)

ans = multiply_and_add(3,6,1,2,2)
```

Define

Example 02.

```
import numpy as np

x = 3,6,1,2,2

ans = np.prod(x), sum(x)
```

Example 02.

```
import numpy as np

def multiply_and_add(*x):
    return np.prod(x), sum(x)

ans = multiply_and_add(3,6,1,2,2)
```

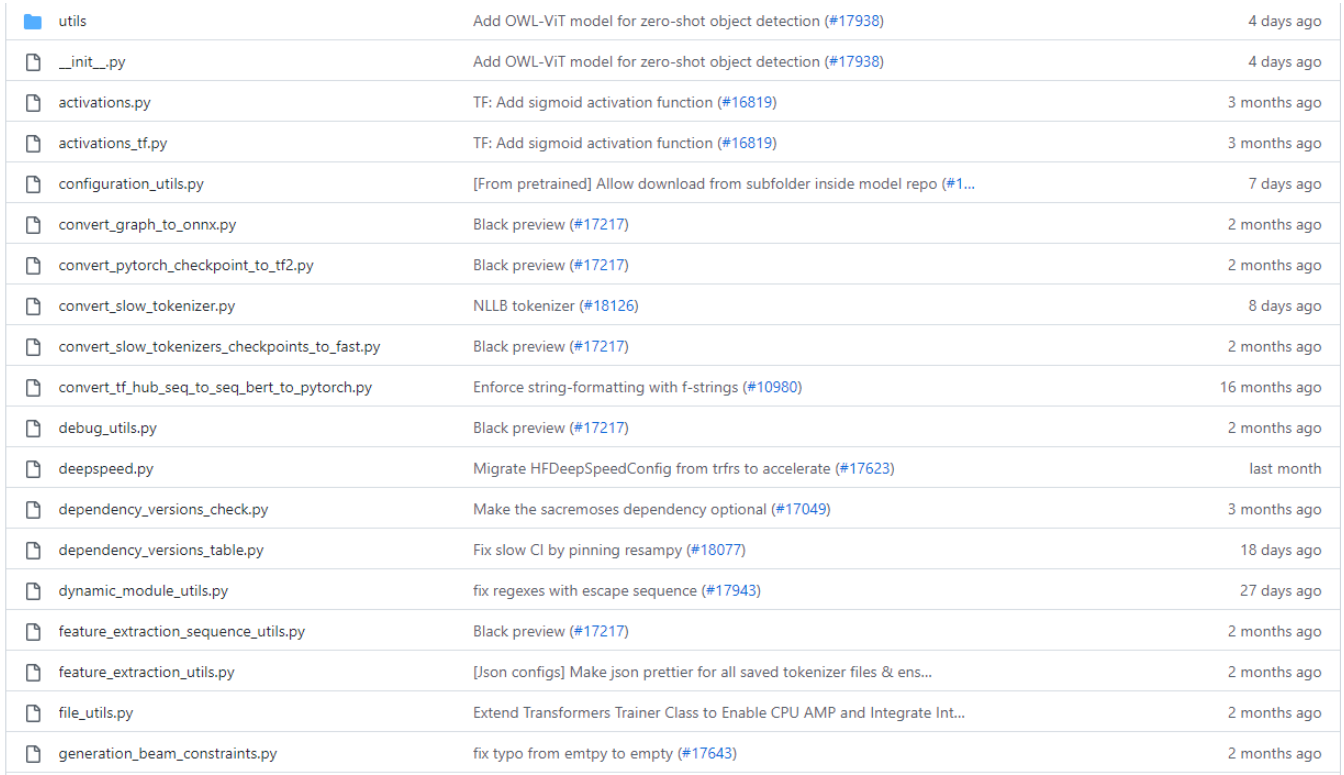
1. 메인 코드의 간소화
2. 타인에게 전달되는 정보량 증가

Class

1. 메인 코드의 간소화
2. 타인에게 전달되는 정보량 증가
3. 확장성 및 재생산성 증가

```
from transformers import PreTrainedModel

backbone = PreTrainedModel(…)
```



utils	Add OWL-ViT model for zero-shot object detection (#17938)	4 days ago
__init__.py	Add OWL-ViT model for zero-shot object detection (#17938)	4 days ago
activations.py	TF: Add sigmoid activation function (#16819)	3 months ago
activations_tf.py	TF: Add sigmoid activation function (#16819)	3 months ago
configuration_utils.py	[From pretrained] Allow download from subfolder inside model repo (#1...	7 days ago
convert_graph_to_onnx.py	Black preview (#17217)	2 months ago
convert_pytorch_checkpoint_to_tf2.py	Black preview (#17217)	2 months ago
convert_slow_tokenizer.py	NLLB tokenizer (#18126)	8 days ago
convert_slow_tokenizers_checkpoints_to_fast.py	Black preview (#17217)	2 months ago
convert_tf_hub_seq_to_seq_bert_to_pytorch.py	Enforce string-formatting with f-strings (#10980)	16 months ago
debug_utils.py	Black preview (#17217)	2 months ago
deepspeed.py	Migrate HFDeepSpeedConfig from trfrs to accelerate (#17623)	last month
dependency_versions_check.py	Make the sacremoses dependency optional (#17049)	3 months ago
dependency_versions_table.py	Fix slow CI by pinning resampy (#18077)	18 days ago
dynamic_module_utils.py	fix regexes with escape sequence (#17943)	27 days ago
feature_extraction_sequence_utils.py	Black preview (#17217)	2 months ago
feature_extraction_utils.py	[Json configs] Make json prettier for all saved tokenizer files & ens...	2 months ago
file_utils.py	Extend Transformers Trainer Class to Enable CPU AMP and Integrate Int...	2 months ago
generation_beam_constraints.py	fix typo from emtpy to empty (#17643)	2 months ago

Class – Example

```
class simple_calculator:
    def __init__(self):
        print('Simple Calculator Obj. created')

    def add(self, x, y):
        return x+y

    def subtract(self, x, y):
        return x-y

    def multiply(self, x, y):
        return x * y
```

나누기는?

```
class simple_calculator:
    def __init__(self):
        print('Simple Calculator Obj. created')

    def add(self, x, y):
        return x+y

    def subtract(self, x, y):
        return x-y

    def multiply(self, x, y):
        return x * y

    def divide(self, x, y):
        return x / (y+1e-10)
```

Class – Example

```
class simple_calculator:
    def __init__(self):
        print('Simple Calculator Obj. created')

    def add(self, x, y):
        return x+y

    def subtract(self, x, y):
        return x-y

    def multiply(self, x, y):
        return x * y
```

```
class modified_calculator(simple_calculator):
    def __init__(self):
        super().__init__()
        print('Modified Calculator Obj. created')

    def divide(self, x, y):
        return x/(y+1e-10)
```

확장성 및 재생산성 증가

Class – Example

생성할 때 parameter가 주어진다면?

```
class simple_calculator:
    def __init__(self,name):
        self.name = name
        print(f'{name} Obj. created')

    def add(self, x, y):
        return x+y

    def subtract(self, x, y):
        return x-y

    def multiply(self, x, y):
        return x * y
```

```
class modified_calculator(simple_calculator):
    def __init__(self):
        super().__init__()
        print('Modified Calculator Obj. created')

    def divide(self, x, y):
        return x/(y+1e-10)
```

Class – Example

생성할 때 parameter가 주어진다면?

```
class simple_calculator:
    def __init__(self,name):
        self.name = name
        print(f'{name} Obj. created')

    def add(self, x, y):
        return x+y

    def subtract(self, x, y):
        return x-y

    def multiply(self, x, y):
        return x * y
```

```
class modified_calculator(simple_calculator):
    def __init__(self):
        super().__init__()
        print('Modified Calculator Obj. created')

    def divide(self, x, y):
        return x/(y+1e-10)
```

Class – Example

생성할 때 parameter가 주어진다면?

```
class simple_calculator:
    def __init__(self,name):
        self.name = name
        print(f'{name} Obj. created')

    def add(self, x, y):
        return x+y

    def subtract(self, x, y):
        return x-y

    def multiply(self, x, y):
        return x * y
```

```
class modified_calculator(simple_calculator):
    def __init__(self,name):
        super().__init__(name)
        print('Modified Calculator Obj. created')

    def divide(self, x, y):
        return x/(y+1e-10)
```

보다 상위 클래스를 위한 super의 paramete를 채워줘야 함.

따라서 double asterisk를 활용한 dictionary로 하위 클래스를 활용하게 됨.

Class – Example

```
parser = argparse.ArgumentParser()

parser.add_argument('--scaler_str', default='Standard', type=
parser.add_argument('--reg_str', default='RF', type=str, help
parser.add_argument('--training', default=True, type=bool, h
parser.add_argument('--model_dir_path', default=os.path.join(
parser.add_argument('--path', default='', type = str, help=''
parser.add_argument('--targets', default='A', type=str, help=
parser.add_argument('--name', default='', type=str, help='')

args = parser.parse_args('')
```

```
In [3]: args.__dict__
```

```
Out[3]: {'scaler_str': 'Standard',
 'reg_str': 'RF',
 'training': True,
 'model_dir_path': './ckpt',
 'path': '',
 'targets': 'A',
 'name': ''}
```

```
REG = regressor(**args.__dict__)
REG(train_df)
y_hat = REG.__predict__(valid_df)
y_real = valid_df['Flute_'+args.targets].values
```

```
class regressor:
    def __init__(
        self,
        scaler_str = None,
        reg_str = None,
        training = True,
        model_dir_path = None,
        path = None,
        targets = None,
        name = None,
    ):

        self.targets = targets
        assert self.targets in ['A','B'], 'Choose one of the targets "A" or "B"'

        self.training = training
        if not self.training:
            assert path != None, #
            'Must filled of argument "path" when the training = False'

        if self.training:
            self.model_path = os.path.join(model_dir_path,path)
            os.makedirs(self.model_path,511,True)
            self.x_scaler = self.__set_scaler__(scaler_str)
            self.y_scaler = self.__set_scaler__(scaler_str)
            self.reg = self.__set_reg__(reg_str)
```

Q & A